1 实验内容及目的

完成小车的车牌定位与识别。

2 实验相关原理描述

2.1 边缘检测

本次实验中使用的方法是Roberts算子。其原理基于两个简单的2×2滤波器，分别作用于原始图像的水平和垂直方向，得到对应的水平和垂直方向的差分值，然后通过对这两个差分值进行平方和再开根的操作得到最终的边缘检测结果。

具体地，Roberts算子通过对输入图像的每一个像素点进行如下操作来检测边缘：

1. 将当前像素点及其相邻的右下角像素点（即2×2的区域）分别用两个2×2的滤波器进行卷积计算，得到相应的水平和垂直方向的差分值。

2. 将得到的水平和垂直方向的差分值进行平方和操作，然后再开根，得到最终的边缘强度值。

3. 如果计算得到的边缘强度值超过预先设定的阈值，则将当前像素标记为边缘点，否则标记为非边缘点。

通过使用Roberts算子，可以非常快速地检测出图像中的边缘，而且对于图像中出现的一些细节和小型边缘也能够进行有效的检测。

2.2 开闭运算

对于边缘图像可以使用开闭运算的操作将其进行形态学的处理。这样能够让图像中的噪点和不必要的边缘被去除，同时也会将断开的边缘连接起来，使得边缘线条更加完整和清晰。

开运算：先进行腐蚀操作，再进行膨胀操作，可去除图像中的小型边缘和噪点，同时减少边缘厚度和长度。开运算可以表达为：开运算(A， B) = 膨胀(腐蚀(A， B))

闭运算：先进行膨胀操作，再进行腐蚀操作，可以填补图像中的空隙、连接断点，并增加边缘的厚度和长度。闭运算可以表达为：闭运算(A， B) = 腐蚀(膨胀(A， B))

其中，A表示输入的图像，B是指定的结构元素。结构元素是一个定义了类似卷积核的形状，用于形态学变换中的模板匹配。

2.3 矩形化

对开闭运算后图片中的非矩形区域进行填充，便于后续使用图片属性识别出车牌。

2.4 图像分割

2.4.1 基于全局的阈值分割(大津法)

基于全局的阈值分割的基本思想是确定一个阈值，将图像中的像素分为两类：一类是满足阈值条件的前景像素，另一类是不满足条件的背景像素。大津法是全局阈值分割的一种常用算法，其原理如下：

1.首先，需要确定一个合适的阈值T。

2.将图像中的所有像素分为两类，一类是灰度值小于等于阈值T的前景像素，一类是灰度值大于阈值T的背景像素。

3.计算两类像素的像素数目、像素值的均值和方差等参数。

4.根据类内方差与类间方差之比来评估当前阈值的分割质量，此值越大，则表示分割效果越好。

5.重复步骤2~4，分别计算不同阈值下的类内方差与类间方差之比，并选取能得到最大类间方差之比的阈值作为分割阈值。

通过大津法得到的分割阈值，可以将图像中的前景和背景像素分离开来，从而实现对图像的分割。基于全局的阈值分割算法具有简单易用、计算量小等优点，常用于对灰度图像进行分割，应用广泛。

2.4.2 基于局部的阈值分割(bernsen)

bernsen算法的原理是，对于待分割图像中的每一个像素，都计算其所在局部区域内的像素的最大灰度值与最小灰度值，并求出两者之差，作为当前像素的阈值。如果该区域内灰度值差异较大，则可以认为该区域包含多种纹理。这时应该使用较大的阈值来作为分割阈值，将图像中的纹理分块。反之，如果该区域内灰度值差异较小，则应该使用较小的阈值进行分割，以使图像中的纹理得到保留。

具体步骤分解如下：

1. 对于待分割图像中的每一个像素，都选取一个大小为N×N的局部区域。
2. 在该区域内找到灰度值的最大值max和最小值min，计算阈值V。
3. 将像素点的灰度值与阈值V进行比较，大于等于阈值的像素归为前景，否则为背景。
4. 重复上述步骤，直到对图像中的所有像素进行了标记。

基于局部的阈值分割方法相对于基于全局的阈值分割方法来说，更加灵活、自适应，能够有效处理含有多种纹理的图像。但要注意的是，由于该算法是基于局部区域进行阈值计算的，所以分割结果可能会受到噪声或者图像亮度变化的影响。因此，在实际应用中，我们需要根据具体问题的需要，选取合适的算法和参数来进行图像的分割。

2.5 余弦相似度

余弦相似度是一种常用的相似度度量方法，适用于在数据挖掘、信息检索、图像处理、自然语言处理等领域的相似性比较问题。余弦相似度是通过计算两个向量之间的夹角余弦值来衡量它们之间的相似程度。

余弦相似度的测量结果在-1到1之间，当结果为1时，表示两个向量在方向上完全相同，在空间直线上投影时，其夹角为0°；当结果为-1时，则表示两个向量在方向上完全相反，在空间直线上投影时，其夹角为180°。当结果为0时，则表示两个向量在方向上互相垂直，在空间直线上投影时，其夹角为90°。

3 实验过程

3.1 车牌定位

3.1.1 边缘检测

由于车牌定位的任务是找出车牌的位置，故此时的边缘检测不需要保留过多的细节信息，所以我没有使用Canny算子，而使用了Roberts算子。

```
img = rgb2gray(imread('test1.jpeg'));

% 边缘检测，使用roberts算子
img_edge = edge(img,'roberts');
```

以下是检测效果：

3.1.2 开闭运算(对边缘检测图先开后闭)

先定义矩形结构元素，再使用imopen()和imclose()函数对边缘检测图进行开闭运算，之后再删去过小的连通域。

```
% 开闭运算
se = strel("rectangle",[10 20]);
open_img = imopen(inverted_img,se);
close_img = imclose(open_img,se);

% 标记连通域
labeled_img = bwlabel(close_img);

% 获取连通域的属性
stats = regionprops(labeled_img, 'Area');

% 设定阈值，用于删除过小的连通域
threshold = 1000;

% 删除过小的连通域
for i = 1:length(stats)
    if stats(i).Area < threshold
        labeled_img(labeled_img == i) = 0;
    end
end

labeled_img = imbinarize(labeled_img);
```

以下是开闭运算处理后的结果：

3.1.3 矩形化

将不规则的连通区域矩形化，方便后续检测车牌。

其具体步骤如 下:1若 G(i，j)为黑点，而且它的 4-邻域中有至少任意3个像素为白点，则令其为白点;2若 G(i，j)为白点，而且它的4-邻域中至少2个像素为黑点，则令其为黑点。

该步骤要从(0,0)(m,0)(0,n)(m,n)四个起始点对图像做四次循环迭代操作。**每次的循环都是先找出目标像素的上下左右的4-邻域像素，然后根据判断条件对目标像素进行赋值。**

```
% 获取尺寸
[m, n] = size(labeled_img);

% 遍历图像中的每个像素（左上角）
for i = 2:m-1
    for j = 2:n-1
        % 获取当前像素的值
        pixel = labeled_img(i, j);

        % 获取4-邻域的像素值
```

```
        neighbors = [labeled_img(i-1, j), labeled_img(i+1, j), labeled_img(i,
j-1),labeled_img(i, j+1)];

        % 统计4-邻域中的黑色像素数量
        black_count = sum(neighbors(:) == 0);

        % 统计4-邻域中的白色像素数量
        white_count = sum(neighbors(:) == 1);

        if pixel == 0 % 当前像素为黑色
            if white_count > 2 % 如果至少有三个邻域像素为白色
                labeled_img(i, j) = 1; % 将当前像素设置为白色
            end
        else % 当前像素为白色
            if black_count >= 2 % 如果至少有两个邻域像素都为黑色
                labeled_img(i, j) = 0; % 将当前像素设置为黑色
            end
        end
    end
end

% 右上角
for i = m-1:-1:2
    for j = 2:n-1
        pixel = labeled_img(i, j);
        neighbors = [labeled_img(i-1, j), labeled_img(i+1, j), labeled_img(i,
j-1),labeled_img(i, j+1)];
        black_count = sum(neighbors(:) == 0);
        white_count = sum(neighbors(:) == 1);

        if pixel == 0 % 当前像素为黑色
            if white_count > 2 % 如果至少有三个邻域像素为白色
                labeled_img(i, j) = 1; % 将当前像素设置为白色
            end
        else % 当前像素为白色
            if black_count >= 2 % 如果至少有两个邻域像素都为黑色
                labeled_img(i, j) = 0; % 将当前像素设置为黑色
            end
        end
    end
end

% 左下角
for i = 2:m-1
    for j = n-1:-1:2
        pixel = labeled_img(i, j);
        neighbors = [labeled_img(i-1, j), labeled_img(i+1, j), labeled_img(i,
j-1),labeled_img(i, j+1)];
        black_count = sum(neighbors(:) == 0);
        white_count = sum(neighbors(:) == 1);
        if pixel == 0 % 当前像素为黑色
            if white_count > 2 % 如果至少有三个邻域像素为白色
```

```
                labeled_img(i, j) = 1; % 将当前像素设置为白色
            end
        else % 当前像素为白色
            if black_count >= 2 % 如果至少有两个邻域像素都为黑色
                labeled_img(i, j) = 0; % 将当前像素设置为黑色
            end
        end
    end
end

% 右下角
for i = m-1:-1:2
    for j = n-1:-1:2
        pixel = labeled_img(i, j);
        neighbors = [labeled_img(i-1, j), labeled_img(i+1, j), labeled_img(i,
j-1),labeled_img(i, j+1)];
        black_count = sum(neighbors(:) == 0);
        white_count = sum(neighbors(:) == 1);
        if pixel == 0 % 当前像素为黑色
            if white_count > 2 % 如果至少有三个邻域像素为白色
                labeled_img(i, j) = 1; % 将当前像素设置为白色
            end
        else % 当前像素为白色
            if black_count >= 2 % 如果至少有两个邻域像素都为黑色
                labeled_img(i, j) = 0; % 将当前像素设置为黑色
            end
        end
    end
end
```

以下是矩形化后的效果图，可以看出，车牌的位置已经清晰可见，下一步就是把车牌所在的矩形区域筛选并标记出来。

3.1.4 选择区域

对上列矩形区域进行筛选，对于车牌，我们利用如下约束进行筛选:

1. 矩形长宽比

   ```
   % 矩形长宽比阈值
   ratio = [2, 8];
   ```

2. 区域蓝色像素点所占比例(针对国内车牌为蓝底)

   ```
   % 蓝色像素比例阈值
   blue_ratio = 0.5;

   % 像素为蓝色的条件
   if hsv_img(y, x, 1) <= 0.6667 && hsv_img(y, x, 1) >= 0.5833
     blue_count = blue_count + 1;
   end
   ```

3. 候选区域垂直投影函数与其 y=平均值的这条直线交点的个数要在一定阈值内， 这里选择的区间是15到45。

```
        % 交点个数阈值
        count = [15, 45];
```

以下是选择区域的代码:

```
% 获取图像的尺寸
[height, width] = size(labeled_img);

% 存储符合条件的车牌位置
plate = [];

% 标记图像中的对象
L = bwlabel(imcomplement(labeled_img));

% 计算区域属性
stats = regionprops(L, 'BoundingBox');

for i = 1:numel(stats)
    % 获取当前矩形区域的边界框信息
    bbox = stats(i).BoundingBox;

    % 计算矩形的长宽比
    wh = bbox(3) / bbox(4);

    % 检查长宽比是否符合约束条件
    if (wh > 2 && wh < 8)
        % 统计矩形区域中蓝色像素点的数量
        blue_count = 0;
        for y = bbox(2)+0.5:bbox(2) + bbox(4) - 0.5
            for x = bbox(1)+0.5:bbox(1) + bbox(3) - 0.5
                if hsv_img(y, x, 1) <= 0.6667 && hsv_img(y, x, 1) >= 0.5833
                    blue_count = blue_count + 1;
                end
            end
        end


        % 计算蓝色像素点的比例
        br = blue_count / (bbox(3) * bbox(4));

        % 检查蓝色像素比例是否符合约束条件
        if br > blue_ratio
            % 计算交点个数
            pj_count = Verticalprojection(img(bbox(2)+0.5:bbox(2) + bbox(4) - 0.5,
bbox(1)+0.5:bbox(1) + bbox(3) - 0.5));

            % 检查交点个数是否符合约束条件
            if pj_count > count(1) && pj_count < count(2)
                % 将符合条件的车牌位置添加到结果中
                plate = [plate; bbox];
            end
        end
```

```
    end
end

% 在原始图像中标记车牌位置
figure,imshow(rgb_img);
hold on;
for i = 1:size(plate, 1)
    rectangle('Position', plate(i, :), 'EdgeColor', 'r', 'LineWidth', 2);
end
hold off;
```

标记出的车牌位置如下：

可见车牌位置已被较为精确地定位。

3.2 图像分割

3.2.1 基于全局的阈值分割(大津法)

根据大津法的一般步骤进行实现：

```
p = plate(1,:);
rgb_plate_img = rgb_img(p(2)+0.5:p(2) + p(4) - 0.5, p(1)+0.5:p(1) + p(3) - 0.5,:);
plate_img = rgb2gray(rgb_img(p(2)+0.5:p(2) + p(4) - 0.5, p(1)+0.5:p(1) + p(3) - 0.5,:));

% 获取灰度图像的直方图
num_pixel = numel(plate_img);
histogram = imhist(plate_img) / num_pixel;

% 计算所有可能的阈值的类间方差，并选择最大值
max_variance = 0;
threshold = 0;

for i = 1:256
    % 计算类别概率
    P1 = sum(histogram(1:i));
    P2 = 1 - P1;

    % 计算均值
    mean_1 = sum((0:i-1)'.*histogram(1:i)) / P1;
    mean_2 = sum((i:255)'.*histogram(i+1:end)) / P2;

    % 计算类间方差
    variance = P1*P2*(mean_1 - mean_2)^2;

    % 更新阈值和最大方差
    if (variance > max_variance)
        max_variance = variance;
        threshold = i-1;
    end
end

% 根据阈值进行图像分割
```

```
segmented_img = plate_img > threshold;
```

3.2.2 基于局部的阈值分割(bernsen)

根据bernsen算法的一般步骤进行实现:

```
p = plate(1,:);
rgb_plate_img = rgb_img(p(2)+0.5:p(2) + p(4) - 0.5, p(1)+0.5:p(1) + p(3) - 0.5,:);
plate_img = rgb2gray(rgb_img(p(2)+0.5:p(2) + p(4) - 0.5, p(1)+0.5:p(1) + p(3) - 0.5,:));

% 定义局部窗口大小和亮度阈值
window_size = 1;  % 窗口大小
brightness_threshold = 50;  % 亮度阈值

% 获取图像尺寸
[rows, cols] = size(plate_img);

% 初始化二值化图像
binary_image = zeros(rows, cols);

% 遍历图像中的每个像素
for i = 1:rows
    for j = 1:cols
        % 获取当前像素的局部窗口
        row_start = max(i - floor(window_size/2), 1);
        row_end = min(i + floor(window_size/2), rows);
        col_start = max(j - floor(window_size/2), 1);
        col_end = min(j + floor(window_size/2), cols);
        window = plate_img(row_start:row_end, col_start:col_end);

        % 计算局部窗口中的最小和最大像素值
        min_val = min(window(:));
        max_val = max(window(:));

        % 计算局部窗口的动态阈值
        threshold = (max_val + min_val) / 2;

        % 检查当前像素的亮度是否超过动态阈值
        if plate_img(i, j) >= threshold - brightness_threshold && plate_img(i, j) <= threshold +
brightness_threshold
            binary_image(i, j) = 255;
        end
    end
end

binary_image  = imcomplement(imbinarize(binary_image));
```

由结果图可以看出，大津法对于背景和目标之间对比度明显的图像有良好的效果，但对于噪声干扰和阴影等问题还需要进行进一步处理，而bernsen算法对于背景和目标之间的对比度不明显的图像具有很好的效果。

3.3 字符分离

这一步较为简单，可以利用栅栏法(即寻找字符间的空隙)分离字符，这里利用y方向上像素和为0来寻找分离点。

```
% 先将图像分割后的图片进行灰度反转
binary_image  = imcomplement(binary_image);

x1 = [];
x2 = [];
X = [];

for x = 1:cols
    if sum(binary_image(:,x)) == 0
        % 保存像素和为0的列
        X = [X, x];
    end
end

for i = 1:length(X)-1
    if X(i) + 1 ~= X(i+1)
        x1 = [x1; X(i)];
        x2 = [x2; X(i+1)];
    end
end

figure;
path = './iden_num/';
for n = 1:length(x1)
    subplot(1,7,n),imshow(rgb_plate_img(1:rows, x1(n):x2(n),:));
    % 将分离后的字符图片储存下来，方便下一步识别
    imwrite(rgb_plate_img(1:rows, x1(n):x2(n)),[path,num2str(n),'.bmp']);
end
```

3.4 车牌识别

这里计算分离得到的字符图片与模版图片之间的余弦相似度，相似度高的即为目标图片。

第一步自定义计算余弦相似度函数cosine_similarity，传入两个二值图，转化为向量，余弦相似度等于向量点积与范数之积的比值。

```
function [similarity] = cosine_similarity(img1, img2)

% 将二值图像转换为向量
vec1 = double(img1(:)');
vec2 = double(img2(:)');

% 计算余弦相似度
similarity = dot(vec1, vec2) / (norm(vec1) * norm(vec2));
end
```

使用自定义函数cosine_similarity将上一步字符分离后保存的图片与模版进行匹配，选择余弦相似度最大的模版作为车牌识别的结果。以下是识别结果：

命令行窗口

```
省份简称：赣.bmp
字符：6.bmp
字符：6.bmp
字符：L.bmp
字符：L.bmp
字符：J.bmp
字符：L.bmp
```