作业9

1. **c++/matlab编程实现 canny 算子进行边缘检测。**

* **源代码**

% 读取图像

img = imread(r'C:\Users\R9000X\Desktop

\ mountains.jpg');

% 转换为灰度图像

grayImg = rgb2gray(img);

% 使用Canny算子进行边缘检测

cannyImg = edge(grayImg, 'Canny');

% 显示原始图像和Canny边缘检测结果

subplot(1, 2, 1);

imshow(grayImg);

title('原始图像');

subplot(1, 2, 2);

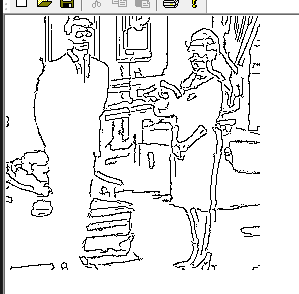
imshow(cannyImg);

title('Canny边缘检测');

% 保存Canny边缘检测结果

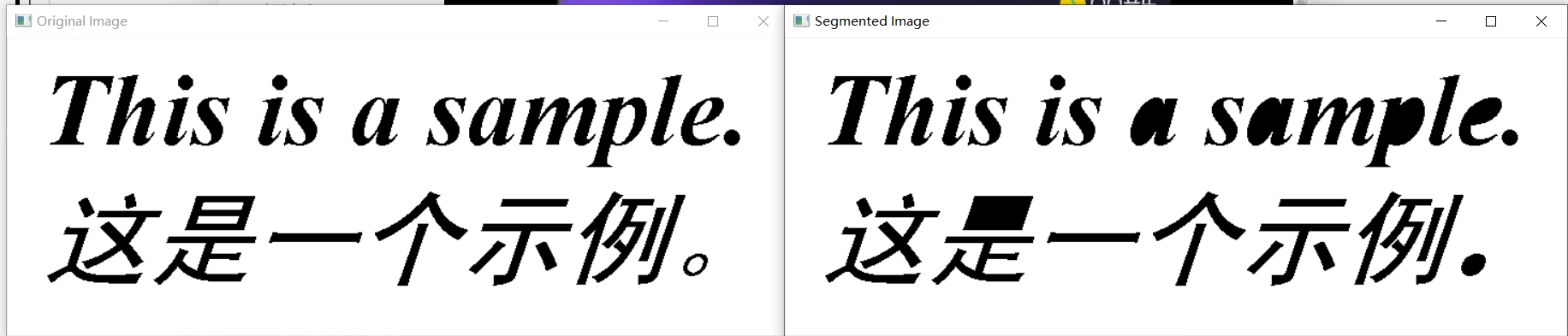
imwrite(cannyImg, 'canny\_output.jpg');

* **实验结果**



1. **编程实现区域生长图像分割。**

* **实验结果：**



* **源代码：**

function segmented = region\_growing(image, seed, threshold)

% 获取图像的尺寸

[height, width] = size(image);

% 创建空的标记图像，与原始图像相同大小，用于记录分割结果

segmented = zeros(height, width);

% 获取种子点的灰度值

seed\_value = image(seed(1), seed(2));

% 定义生长方向（上、下、左、右）

directions = [-1, 1, 0, 0; 0, 0, -1, 1];

% 将种子点添加到生长区域中

segmented(seed(1), seed(2)) = 1;

% 存储待生长的像素点的队列

queue = seed;

while ~isempty(queue)

% 取出当前待生长的像素点

current\_point = queue(1, :);

queue(1, :) = [];

% 遍历当前像素点的邻域

for i = 1:size(directions, 2)

x = current\_point(1) + directions(1, i);

y = current\_point(2) + directions(2, i);

% 判断邻域像素点是否在图像范围内

if x >= 1 && x <= height && y >= 1 && y <= width

% 判断邻域像素点是否已经被访问过

if segmented(x, y) == 0

% 计算邻域像素点与种子点的灰度差值

diff = abs(double(image(x, y)) - double(seed\_value));

% 判断差值是否满足生长条件

if diff <= threshold

% 将邻域像素点标记为属于生长区域

segmented(x, y) = 1;

% 将邻域像素点添加到待生长的队列中

queue = [queue; x, y];

end

end

end

end

end

end

% 读取图像

image = imread('images/test.bmp');

image = rgb2gray(image);

% 设置种子点

seed = [100, 100];

% 设置生长阈值

threshold = 10;

% 进行区域生长分割

segmented\_image = region\_growing(image, seed, threshold);

% 显示原始图像和分割结果

subplot(1, 2, 1);

imshow(image);

title('原始图像');

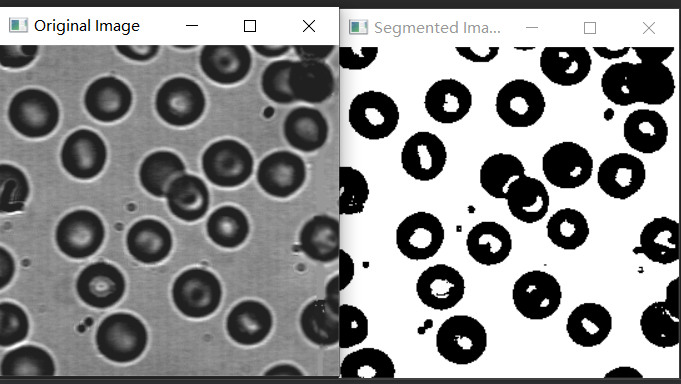
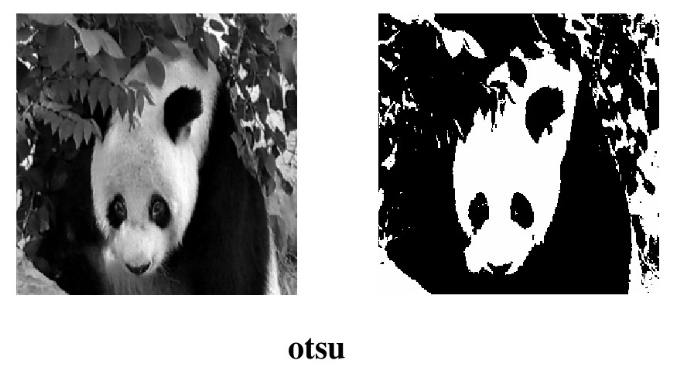
subplot(1, 2, 2);

imshow(segmented\_image);

title('分割结果');

1. **c++/matlab编程实现OTSU方法的图像分割。**

* **实验结果：**



* **源代码：**

% 读取图像

image = imread('input\_image.jpg');

% 将图像转换为灰度图像

gray\_image = rgb2gray(image);

% 使用OTSU方法计算最佳阈值

threshold = graythresh(gray\_image);

% 对图像进行阈值分割（二值化）

segmented\_image = imbinarize(gray\_image, threshold);

% 将低于阈值的像素赋值为0，高于阈值的像素赋值为255

segmented\_image(segmented\_image == 0) = 0;

segmented\_image(segmented\_image == 1) = 255;

% 显示原始图像和分割结果

figure;

subplot(1, 2, 1);

imshow(gray\_image);

title('Original Image (Grayscale)');

subplot(1, 2, 2);

imshow(segmented\_image);

title('Segmented Image (Binary)');

% 将分割结果保存为新的图像

imwrite(segmented\_image, 'segmented\_image.jpg');