数字图像处理实践代码

线段检测源码

```
clear;
clc;
Orig_Img = imread('test.png');
Orig_Img = im2gray(Orig_Img);

% 去除椒盐噪声
filter_img= medfilt2(Orig_Img, [5,5]);

% 提取边界
BW = edge(filter_img, "canny");
% 进行霍夫变换
[H, T, R] = hough(BW, 'RhoResolution', 3.2, 'Theta', -90:89);
P = houghpeaks(H, 5, 'threshold', ceil(0.3*max(H(:))));
lines = houghlines(BW, T, R, P, 'FillGap', 5, 'MinLength', 7);

% 储存所有直线的长度
len_all = zeros(1, length(lines));

max_len = 0;
for k=1:length(lines)
    xy = [lines(k).point1; lines(k).point2];
    % 计算线段的长度
    len = norm(lines(k).point1 - lines(k).point2);
    len_all(:,k) = len;
end

% 找到前几条最长的线段长度
target_len = maxk(len_all, 5);

% 寻找对应的三条线段
for k=1:length(lines)
    xy = [lines(k).point1; lines(k).point2];
    % 计算线段的长度
    len = norm(lines(k).point1 - lines(k).point2);
    for i = 1: length(target_len)
        if target_len(i) == len
            target_lines(:,i) = lines(k);
        end
    end
end

figure(2);
imshow(BW);
```

```
hold on;
for k = 1: length(target_lines)
    xy = [lines(k).point1; lines(k).point2];
    plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'green');
end
```

圆形检测源代码

```
clear;
clc;
Orig_Img = imread('test.png');
Orig_Img = im2gray(Orig_Img);

imshow(Orig_img);

% 确定搜索圆的半径范围
d = drawline;
pos = d.Position;
diffPos = diff(pos);
diameter = hypot(diffPos(1),diffPos(2));
% 搜索半径在[diameter/2-5 diameter/2+5]左右的圆
[centers,radii] = imfindcircles(Orig_img,[int8(diameter/2)-5
int8(diameter/2)+5],'ObjectPolarity','dark', 'Sensitivity',0.96);
imshow(Orig_img);
h = viscircles(centers,radii);
```

矩形检测源代码

以下几份代码为单个矩形的识别程序。

```
clear;
clc;
Orig_Img = imread('test.png');
Orig_Img = im2gray(Orig_Img);

% 去除椒盐噪声
filter_img= medfilt2(Orig_Img, [5,5]);

% 提取边界
BW = edge(filter_img, "canny");
% 进行霍夫变换
[H, T, R, lines] = GetLinesResult(BW, 1.5563, -48:0.5:-26);
[H1, T1, R1, lines1] = GetLinesResult(BW, 1.55, 47:0.5:60);
```

```
[target_lines] = GetTargetLines(lines, 2);
[target_lines1] = GetTargetLines(lines1, 3);
target_lines1 = target_lines1(:,2:3);

figure(2);
imshow(BW);
hold on;
for k = 1: length(target_lines1)
    xy = [target_lines(k).point1; target_lines(k).point2];
    plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'green');

    xy1 = [target_lines1(k).point1; target_lines1(k).point2];
    plot(xy1(:,1), xy1(:,2), 'LineWidth', 2, 'Color', 'green');
end
```

```
function [target_lines] = GetTargetLines(lines, num)
    % 储存所有直线的长度
    len_all = zeros(1, length(lines));

    for k=1:length(lines)
        % 计算线段的长度
        len = norm(lines(k).point1 - lines(k).point2);
        len_all(:,k) = len;
    end

    % 找到2条最长的线段长度
    target_len = maxk(len_all, num);

    % 寻找对应的2条线段
    for k=1:length(lines)
        % 计算线段的长度
        len = norm(lines(k).point1 - lines(k).point2);
        for i = 1: length(target_len)
            if target_len(i) == len
                target_lines(:,i) = lines(k);
            end
        end
    end
end
```

```
function [H, T, R, lines] = GetLinesResult(BW, RhoResolution, Theta)
    [H, T, R] = hough(BW, 'RhoResolution', RhoResolution, 'Theta', Theta);
    P = houghpeaks(H, 5, 'threshold', ceil(0.3*max(H(:))));
    lines = houghlines(BW, T, R, P, 'FillGap', 5, 'MinLength', 7);
end
```

```
import Foundation
```

```swift
struct TrianglePoint {
    var x: Double
    var y: Double
}

func calcuteWidthAndHeight(point1: TrianglePoint, point2: TrianglePoint, point3:
TrianglePoint) -> (width: Double, height: Double) {
    let width = sqrt(pow((point1.x-point2.x), 2) + pow((point1.y-point2.y), 2))
    let height = sqrt(pow((point3.x-point1.x), 2) + pow((point3.y-point1.y), 2))
    return (width, height)
}

func calcuCoordinate(point1: TrianglePoint, point2: TrianglePoint, point3:
TrianglePoint, point4: TrianglePoint) -> (x: Double, y: Double) {
    let middleX1 = (point1.x + point2.x)/2
    let middleY1 = (point1.y + point2.y)/2

    let middleX2 = (point3.x + point4.x)/2
    let middleY2 = (point3.y + point4.y)/2

    let middleX = (middleX1 + middleX2)/2
    let middleY = (middleY1 + middleY2)/2

    return (middleX, middleY)
}

let point1 = TrianglePoint(x: 157, y: 13)
let point2 = TrianglePoint(x: 157, y: 46)
let point3 = TrianglePoint(x: 290, y: 13)
let point4 = TrianglePoint(x: 290, y: 46)

let coor = calcuCoordinate(point1: point1, point2: point2, point3: point3, point4:
point4)

let bounds = calcuteWidthAndHeight(point1: point1, point2: point2, point3: point3)

// 计算矩形边长
print(bounds.width)
print(bounds.height)

// 计算矩形的中心坐标
print(coor.x)
print(coor.y)
```