Homework 7
Circle Finding Algorithm

Student: Yuanyuan Zhao, 2300013077, School of EECS, zhaoyuanyuan@stu.pku.edu.cn
Lecturer: Yisong Chen

Part 1: Circle Finding Algorithm Design

I implemented 2 algorithms to find circles. In algorithm1.mlx, I attempted to design a customized algorithm, which consists of Canny edge detection, curvature-based arc segmentation, RANSAC-inspired circle fitting and metal reflectance validation. However, it turned out unsatisfactory, so I moved on to implement a Hough Transform version with something novel in algorithm2.mlx, which achieves a good performance.

(a) **Customized Algorithm**

(i) Preprocessing (Canny edge detection)

```
if size(img, 3) == 3
    grayImg = rgb2gray(img);
else
    grayImg = img;
end
edges = edge(grayImg, 'canny', p.Results.EdgeThreshold);
```

(ii) Arc Detection

$$\kappa_i = \frac{\|(P_{i-k} - P_i) \times (P_{i+k} - P_i)\|}{\|P_{i-k} - P_i\|^3}$$

```
function curvature = computeCurvature(contour)
    n = size(contour, 1);
    curvature = zeros(n, 1);
    for i = 1:n
        prev = contour(max(1, i-5), :);
        next = contour(min(n, i+5), :);
        vec1 = contour(i,:) - prev;
        vec2 = next - contour(i,:);
        curvature(i) = abs(vec1(1)*vec2(2) - vec1(2)*vec2(1));
    end
    curvature = curvature / max(curvature);
end
```

(iii) Geometric Verification

```
x = pts(:,1); y = pts(:,2);
A = 2 * [x(2)-x(1), y(2)-y(1); x(3)-x(1), y(3)-y(1)];
b = [x(2)^2 - x(1)^2 + y(2)^2 - y(1)^2;
     x(3)^2 - x(1)^2 + y(3)^2 - y(1)^2];
```

```
center = (pinv(A) * b)';
radius = sqrt((x(1)-center(1))^2 + (y(1)-center(2))^2);
```

(b) **Hough Transform Algorithm**

(i) Gauss Smoothing and Canny Edge Detection

At first Canny detection wad deployed directly without any smoothing, and the results turned out of great redundancy, so I add a gauss filter before edge detection.

```
gaussFilter = fspecial('gaussian', filterSize, sigma);
smoothImg = imfilter(grayImg, gaussFilter, 'replicate');
edges = edge(smoothImg, 'canny');
```

(ii) Hough Transform

```
[y, x] = find(edges);
for i = 1:length(x)
    for rIdx = 1:length(radiusRange)
        r = radiusRange(rIdx);
        for theta = 0:360
            thetaRad = deg2rad(theta);
            a = round(x(i) - r * cos(thetaRad));
            b = round(y(i) - r * sin(thetaRad));
            if a > 0 && a <= size(edges, 2)
                        && b > 0
                        && b <= size(edges, 1)
                accumulator(b, a, rIdx) =
                        accumulator(b, a, rIdx) + 1;
            end
        end
    end
end
```

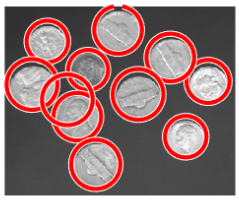
(iii) Find Local Maximum

```
for rIdx = 1:length(radiusRange)
    [b, a] = find(accumulator(:,:,rIdx) >
                0.5 * max(accumulator(:)));
    for i = 1:length(a)
        circles = [circles; a(i), b(i), radiusRange(rIdx)];
    end
end
```

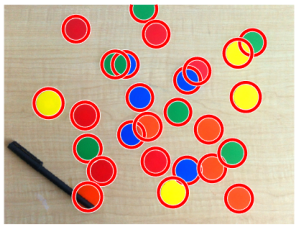
(iv) Remove Redundancy

```
if distance < distanceThreshold
            && radiusDiff < radiusThreshold
```

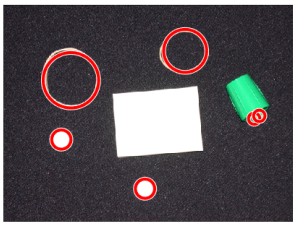
```
                if circles(i, 3) < circles(j, 3)
                    toDelete(i) = true;
                else
                    toDelete(j) = true;
                end
            end
```



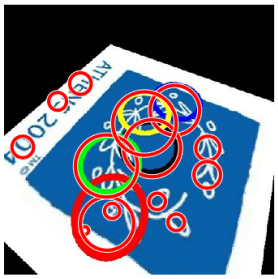
(a) minRadius=25, maxRadius=35



(b) minRadius=15, maxRadius=30



(c) minRadius=10, maxRadius=50



(d) minRadius=5, maxRadius=50

Figure 1: myFindCircle with different radius parameters

Part 2: Result Analysis

The results are satisfactory to a large extent, but there're still some problems. To be specific, in coin.png, there is one redundant circle, and when I adjust the parameter of removeOverlappingCircles function trying to remove it, the correct circle disappears. And in olympic.jpg, the characters and patterns trigger some false circles. Those results show that the curvature algorithm is too sensitive to noise and the selection of redundancy is not perfectly accurate. Thus, the algorithm may not be suitable for some complicated scenario.
In addition, manual adjustment of minRadius and maxRadius is needed, making it inapplicable when the radius of target circles are totally unknown.