```
function detectIrisEdges(imagePath, irisRadiusRange, sensitivity, blurRadius,
searchRadius, stepSize,var)
    % Read the image
    originalImg = imread(imagePath);

    % Check if the image is in grayscale
    if size(originalImg, 3) == 3
        % Convert the image to grayscale if it's in color
        grayImg = rgb2gray(originalImg);
    else
        % Image is already in grayscale
        grayImg = originalImg;
    end

 blurredImg = imgaussfilt(double(grayImg), blurRadius);

    % Apply Canny edge detection
    edges = edge(blurredImg, 'Canny');

   % edges= imsharpen(double(edges));


    % Detect iris circles (inner boundary) using imfindcircles
    [innerCenters, innerRadii, ~] = imfindcircles(edges, irisRadiusRange,
'ObjectPolarity','bright', 'Sensitivity', sensitivity);

    % Assume there is only one inner circle
    innerCenter = innerCenters(1, :);
    innerRadius = innerRadii(1);

    % Initialize parameters for Daugman's integro-differential operator
    maxIrisRadius = 0;
    maxChange = 0;

    % Iterate over possible radii
    for r = searchRadius(1):stepSize:searchRadius(2)
        % Calculate average derivative along the circular path
        avgDerivative = calculateAverageDerivative(edges, innerCenter, r);

        % If the average derivative is greater than the current maximum, update the
maximum radius
        if avgDerivative > maxChange
            maxChange = avgDerivative;
            maxIrisRadius = r;
        end
    end

    disp(innerRadius);
    disp(maxIrisRadius);

%convertToPolar(double(grayImg), innerCenter, innerRadius, maxIrisRadius);
convertToPolarCoordinates(innerCenter, innerRadius, maxIrisRadius,var);


    % Create a new figure for each eye image
    figure;

    % Display the original image with detected iris circles
    imshow(grayImg);
    hold on;

    viscircles(innerCenter, innerRadius, 'EdgeColor', 'b');
    viscircles(innerCenter, maxIrisRadius, 'EdgeColor', 'r');

    % Title the figure with the image filename
    [~, fileName, ~] = fileparts(imagePath);
    title(['Iris Detection - ', fileName]);
end

function avgDerivative = calculateAverageDerivative(img, center, radius)
    % Initialize sum of derivatives
    sumDerivative = 0;

    % Get the size of the image
    [imgHeight, imgWidth] = size(img);

    % Iterate over the circular path
    for theta = 0:0.01:2*pi
        % Calculate coordinates of the point on the circle
        x = round(center(1) + radius * cos(theta));
        y = round(center(2) + radius * sin(theta));

        % Check if the coordinates are within the image dimensions
        if x > 0 && x < imgHeight && y > 0 && y < imgWidth
            % Calculate derivative at the point
            derivative = double(img(x+1, y)) - double(img(x, y));

            % Add the derivative to the sum
            sumDerivative = sumDerivative + derivative;
        end
    end

    % Calculate average derivative
    avgDerivative = sumDerivative / (2 * pi * radius);
end
```