Circle Detection without using Hough Transformations

Elijah Garmon

Department of Computer Science
Florida Polytechnic University, Florida, USA

March 2024

ABSTRACT

This report discusses how I identified circles in various images using OpenCV. As an additional limiting factor, we were not allowed to use the built-in functions of HoughCircle and all other HoughTransformations which can be used to detect the features of images.

CONTENTS

ABSTRACT ii

CONTENTS iii

LIST OF FIGURES iv

LIST OF TABLES v

1 CIRCLE DETECTION 1

2 IMAGE ONE 2

3 IMAGE TWO 4

4 IMAGE THREE 6

REFERENCES 8

LIST OF FIGURES

Figure 2.1 Original Image One 2

Figure 2.2 Detected Circles in Image One 3

Figure 3.1 Original Image Two 4

Figure 3.2 Detected Circles in Image Two 5

Figure 4.1 Original Image Three 6

Figure 4.2 Detected Circles in Image Three 7

LIST OF TABLES

Table 2.1 Centers and Radius' for circles detected in Image One 2

Table 3.1 Centers and Radius' for circles detected in Image Two 4

Table 4.1 Centers and Radius' for circles detected in Image Three 7

1 CIRCLE DETECTION

The same method is used for circle detection for each of these images. The main component for circle detection was the OpenCV function of Contours OpenCV (b) to detect the forming of arcs. Circumference was calculated; if it is 0, it will be ignored and not continue in the program. Afterward, the Area of the arc was calculated.

Next, the circularity was calculated using the equation $4 * \pi * \frac{Circumference^2}{Area}$ and eliminated any circles whose circularity was less than 0.3 and greater than 3. This part of the function determines which partial arcs are removed or included. This method is challenging for image two, as it has many overlapping circles. However, it works the strongest for image one and image three.

Now, the remaining contours were used to form circles and determine their radius and centers. To prevent any leaking of small arcs that were not large circles from filtered background details, only circles whose radius was at least 20 pixels were allowed through. Finally, the circles were drawn onto the original image and sent their centers and radius to the terminal.

2 IMAGE ONE

Figure 2.1. Original Image One

For the first image, multiple filters were applied to leave only the edges. The image was first reduced to grayscale. It was then blurred with a median blur with a k size of 5. Afterward, an edge detection filter was applied using Canny OpenCV (a). The detected circles are listed below:

Center	Radius
(106, 380)	34
(277, 373)	43
(481, 339)	55
(129, 284)	27
(335, 246)	66
(526, 198)	43
(399, 108)	45
(181, 141)	84

Table 2.1. Centers and Radius' for circles detected in Image One

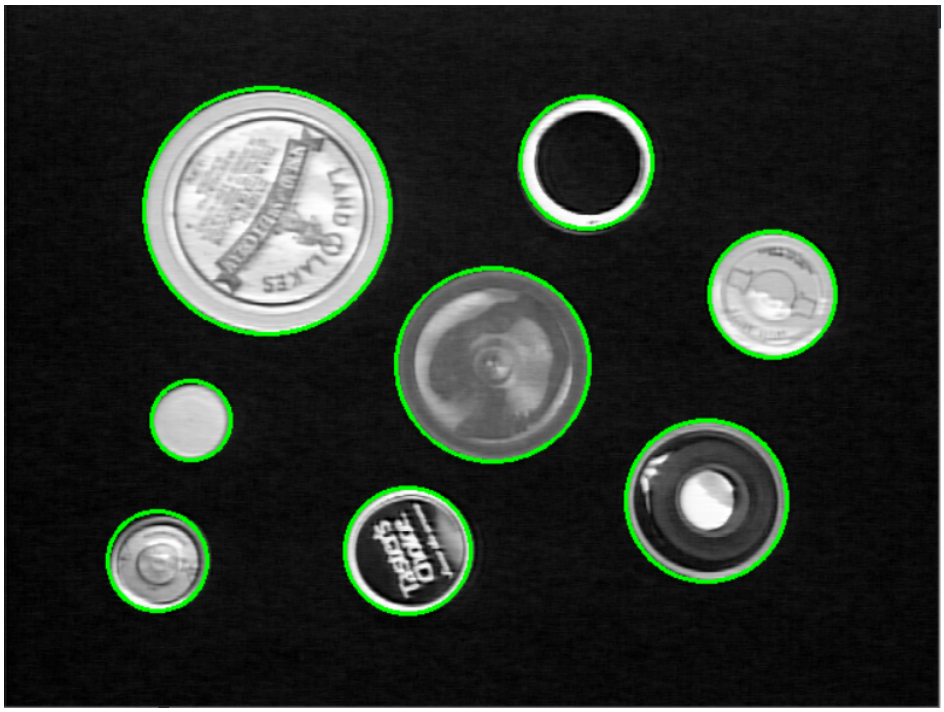

Figure 2.2. Detected Circles in Image One

3 IMAGE TWO

Figure 3.1. Original Image Two

Image Two was the most intricate for the program to detect circles on. Due to its overlapping nature, edge detection needed help with many distinguishing features. Similar to image one, the first step was applying a gray filter. Afterward, a median blur with a k size of 5 was applied. Unlike image one, threshold edge detectionOpenCV (c) was applied instead of canny edge detection to assist with the overlapping. Unfortunately, the best-obtained results only gave two circles. This problem is a limitation of the method used for circle detection. Listed below are the coordinates and circles detected:

Center	Radius
(262, 289)	49
(270, 127)	63

Table 3.1. Centers and Radius' for circles detected in Image Two

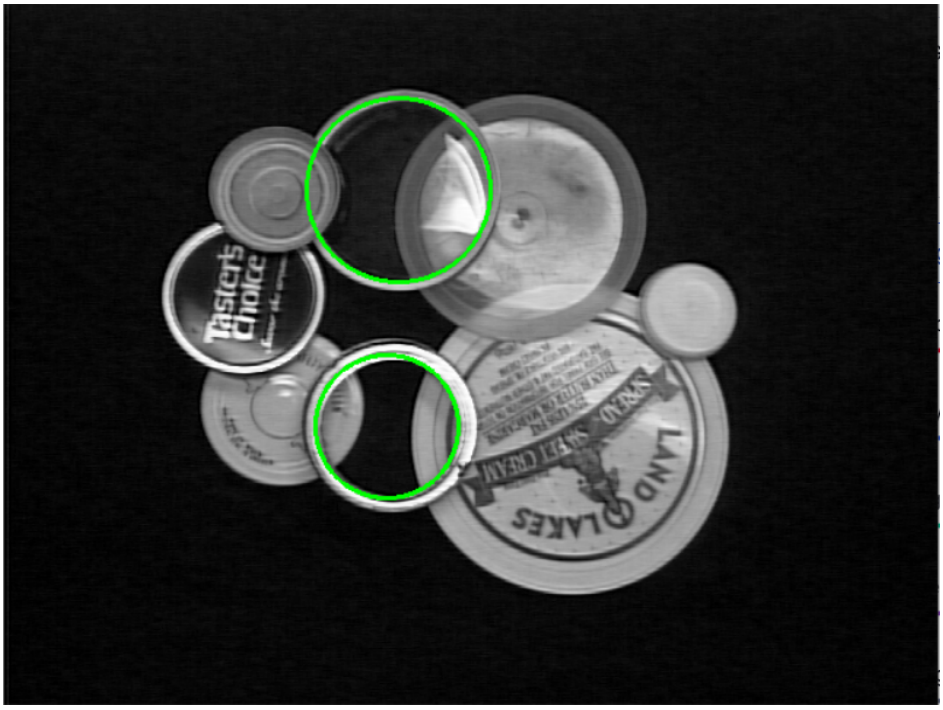

Figure 3.2. Detected Circles in Image Two

4 IMAGE THREE

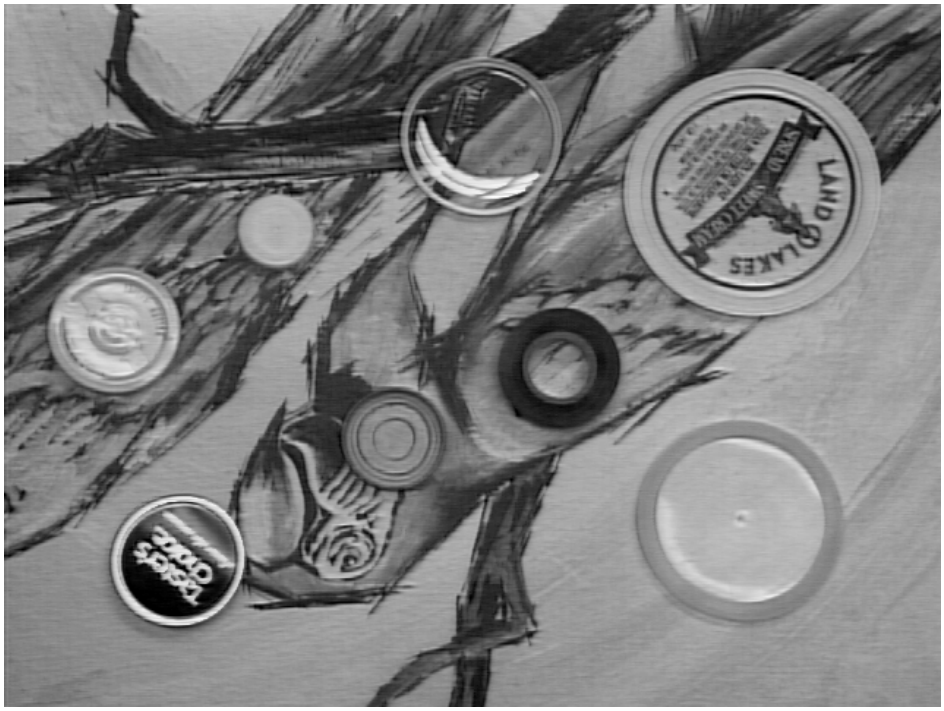

Figure 4.1. Original Image Three

Despite the complexity of image three, this program had an easier time detecting circles since the circles were spread out from each other. Similar to both images before, a grayscale and a median blur with a k size of 5 were applied to image three. Afterward, both methods of using Canny for edge detection and Thresholds for edge detection were applied separately. The circles in both images were combined and applied to the original image. Those coordinates and detected circles can be found below:

Center	Radius
(118, 382)	39
(141, 360)	25
(264, 299)	31
(151, 201)	25
(192, 150)	33
(501, 352)	58
(378, 249)	21
(511, 128)	70

Table 4.1. Centers and Radius' for circles detected in Image Three

Figure 4.2. Detected Circles in Image Three

REFERENCES

OpenCV. “Canny edge detection, <https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html>.

OpenCV. “Contours : Getting started, <https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html>.

OpenCV. “Thresholding, <https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html>.