Circle Detection and Tracking using OpenCV Library

Muhammad Kamal Nadjieb
Computer Science/Informatics Study Program, School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
E-mail: 13514054@std.stei.itb.ac.id

***Abstract*— OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. In this project, the Author will try to detect and track a circle in an image using OpenCV library with the Author's proposed method. The quality of the method is very fine and the noises are low.**

***Keywords*—OpenCV; Thresholding Operations; Image Morphology; Canny; Hough Transforms; HSV;**

I. Introduction

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.

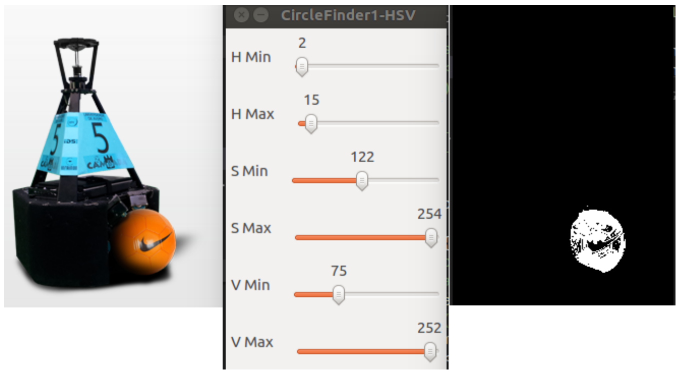
In this project, the Author will try to detect and track a circle in an image using OpenCV library with the Author's proposed method.

II. Literature Study

A. *Thresholding Operations*

According to Bradski and Kaehler (2008), thresholding is the simplest method of image segmentation. Example of its application is to separate out regions of an image corresponding to objects which we want to analyze. To differentiate the pixels we are interested in from the rest (which will eventually be rejected), we perform a comparison of each pixels intensity value with respect to a threshold (determined according to the problem to solve). Once we have separated properly the important pixels, we can set them with a determined value to identify them (i.e. we can assign them a value of 0 (black), 255 (white) or any value that suits our needs).

In this project, the Author will use minimum and maximum HSV intensity values as the thresholds. With OpenCV, we can use cv::inRange function to do that.



H Min 2
H Max 15
S Min 122
S Max 254
V Min 75
V Max 252

Figure 1 Thresholding a RGB frame to HSV frame
Source: Private Documentation

B. *Image Morphology*

According to Bradski and Kaehler (2008), morphological operation is a set of operations that process images based on shapes. Morphological operations apply a structuring element to an input image and generate an output image. The most basic morphological operations are Erosion and Dilation. They have a wide array of uses like removing noise, isolation of individual elements and joining disparate elements in an image, and finding of intensity bumps or holes in an image.

1) *Dilation*

According to Bradski and Kaehler (2008), this operations consists of convoluting an image A with some kernel (B), which can have any shape or size, usually a square or circle. The kernel B has a defined anchor point, usually being the center of the kernel. As the kernel B is scanned over the image, we compute the maximal pixel value overlapped by B and replace the image pixel in the anchor point position with that maximal value. As we can deduce, this maximizing operation causes bright regions within an image to "grow". With OpenCV, we can use cv::dilate function to do that.

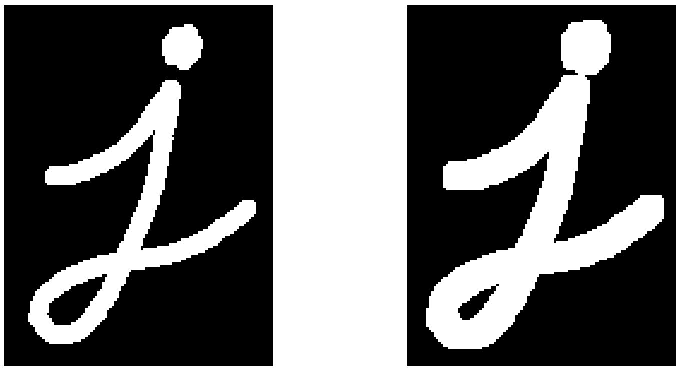


Figure 2 Dilate a frame
Source:
http://docs.opencv.org/2.4/_images/Morphology_1_Tutorial_Theory_Dilatation_2.png

2) Erotion

According to Bradski and Kaehler (2008), this operation is the converse operation of dilation. What this does is to compute a local minimum over the area of the kernel. As the kernel B is scanned over the image, we compute the minimal pixel value overlapped by B and replace the image pixel under the anchor point with that minimal value. With OpenCV, we can use cv::erode function to do that.

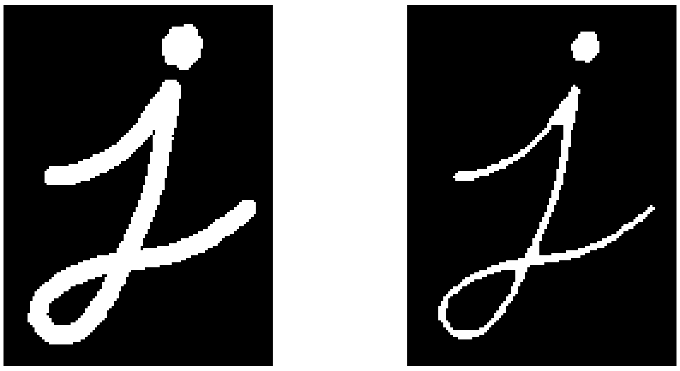


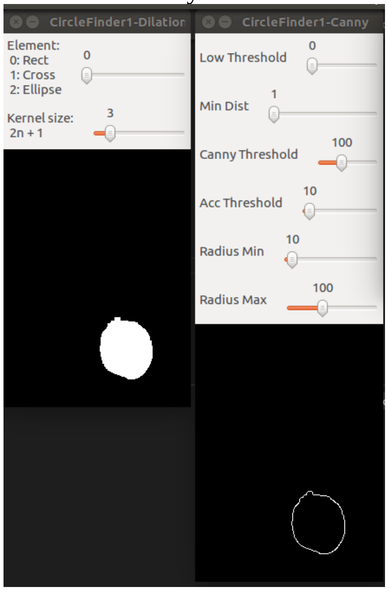
Figure 3 Erode a frame
Source:
http://docs.opencv.org/2.4/_images/Morphology_1_Tutorial_Theory_Erosion_2.png

C. *Canny*

According to Bradski and Kaehler (2008), the Canny Edge detector was developed by John F. Canny in 1986. Also known to many as the optimal detector, Canny algorithm aims to satisfy three main criteria:

1. Low error rate : Meaning a good detection of only existent edges.
2. Good localization : The distance between edge pixels detected and real edge pixels have to be minimized.
3. Minimal response : Only one detector response per edge.

In the Canny algorithm, the first derivatives are computed in x and y and then combined into four directional derivatives. The points where these directional derivatives are local maxima are then candidates for assembling into edges. With OpenCV, we can use cv::Canny to do that.


CircleFinder1-Dilatior

Element: 0
0: Rect
1: Cross
2: Ellipse

Kernel size: 3
2n + 1

CircleFinder1-Canny

Low Threshold 0
Min Dist 1
Canny Threshold 100
Acc Threshold 10
Radius Min 10
Radius Max 100

Figure 4 Canning a frame
Source: Private Documentation

D. *Hough Transforms*

According to Bradski and Kaehler (2008), the Hough transform is a method for finding lines, circles, or other simple forms in an image. The original Hough transform was a line transform, which is a relatively fast way of searching a binary image for straight lines. The transform can be further generalized to cases other than just simple lines.

1) Hough Line Transform

According to Bradski and Kaehler (2008), the Hough Line Transform is a transform used to detect straight lines. To apply the Transform, first an edge detection pre-processing is desirable. In general, a line can be detected by finding the number of intersections between curves. The more curves intersecting means that the line represented by that intersection have more points. In general, we can define a threshold of the minimum number of intersections needed to detect a line. This is what the Hough Line Transform does. It keeps track of the intersection between curves of every point in the image. If the number of intersections is above some threshold, then it declares it as a line with the parameters (r,θ) of the intersection point. With OpenCV, we can use cv::HoughLines to do that.

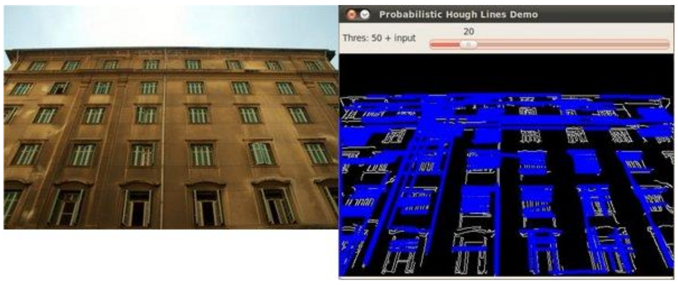


Figure 5 Finding Lines using Hough Line Transform
Source:
http://docs.opencv.org/2.4/_images/Hough_Lines_Tutorial_Original_Image.jpg
http://docs.opencv.org/2.4/_images/Hough_Lines_Tutorial_Result.jpg

2) *Hough Circle Transform*

According to Bradski and Kaehler (2008), the Hough Circle Transform works in a roughly analogous way to the Hough Line Transform. In the line detection case, a line was defined by two parameters (r, θ). With the circle case, we need three parameters to define a circle:

$$C : (x_{center}, y_{center}, r)$$

where (x_{center}, y_{center}) define the center position (gree point) and r is the radius, which allows us to completely define a circle. For sake of efficiency, OpenCV implements a detection method slightly trickier than the standard Hough Transform: The Hough gradient method. In OpenCV, we can use cv::HoughCircles to do that.

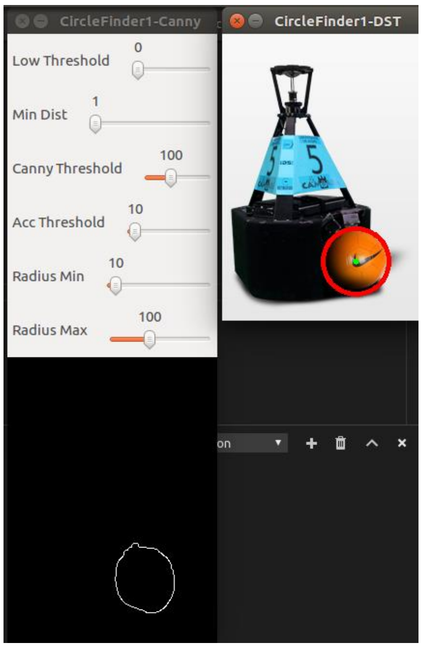

CircleFinder1-Canny
Low Threshold 0
Min Dist 1
Canny Threshold 100
Acc Threshold 10
Radius Min 10
Radius Max 100

Figure 6 Find a circle using Hough Circle Transform
Source: Private Transformation

III. The Proposed Method

The proposed method is follows (the block diagram of the proposed method is shown Figure 7):

1. Capture video or load an image and convert it to RGB frame.
2. Convert the RGB frame to HSV frame.
3. Thresholding the HSV frame with minimum and maximum HSV values
4. Eroding the HSV frame and convert it to erode frame
5. Dilate the erode frame and convert it to dilate frame
6. Find edges from dilate frame and convert it to edges frame using Canny Edges detector
7. Find the circles from edges frame using Hough Circle Transform

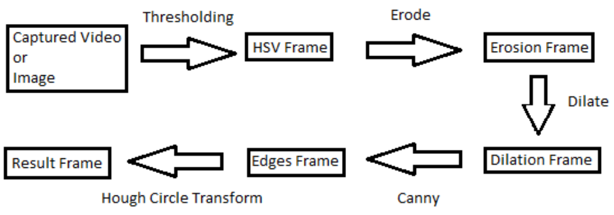

$$\begin{tikzcd}
\boxed{\text{Captured Video or Image}} \arrow[r, "\text{Thresholding}"] & \boxed{\text{HSV Frame}} \arrow[r, "\text{Erode}"] & \boxed{\text{Erosion Frame}} \arrow[d, "\text{Dilate}"] \\
\boxed{\text{Result Frame}} & \boxed{\text{Edges Frame}} \arrow[l, "\text{Hough Circle Transform}"'] & \boxed{\text{Dilation Frame}} \arrow[l, "\text{Canny}"']
\end{tikzcd}$$

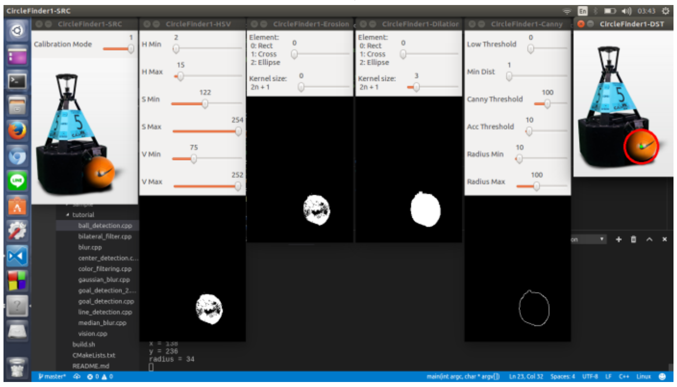
Figure 7 The Proposed Method
Source: Private Documentation

IV. Results and Discussion

The Author have performed some experiments to see if the method can detect a circle in an image or not. The Author use an image in PNG format and video capture from webcam as the test images.

1) *PNG Image Test*

Below is the result of the test.


CircleFinder1-SRC
Calibration Mode 1
CircleFinder1-HSV
H Min 2
H Max 15
S Min 122
S Max 254
V Min 75
V Max 252
CircleFinder1-Erosion
Element: 0: Rect 1: Cross 2: Ellipse 0
Kernel size: 2n + 1 0
CircleFinder1-Dilation
Element: 0: Rect 1: Cross 2: Ellipse 0
Kernel size: 2n + 1 3
CircleFinder1-Canny
Low Threshold 0
Min Dist 1
Canny Threshold 100
Acc Threshold 10
Radius Min 10
Radius Max 100

```
x = 138
y = 236
radius = 34
```

Figure 8 The result of PNG Image Test
Source: Private Documentation

As we can see that the method can detect and track a circle in an image.

2) *Webcam Source Image Test*

Below is the result of the test.

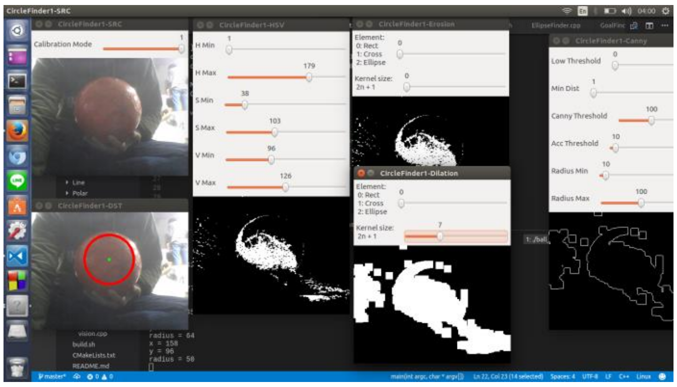


Figure 9 The result of Webcam Source Image Test
Source: Private Documentation

As we can see that the method can detect and track a circle in a captured video from webcam.

Conclusion

OpenCV library is truly a powerful library to do image processing like detect and track a circle in an image. The quality of the method is very fine and the noises are low. The method can be use to detect and track a ball in a soccer robot competition in KRI (Kontes Robot Indonesia).

Acknowledgment

The author would like to thank Allah SWT for His guidance since the Author started making this paper until it is finally done. The author would also express his gratitude to his three lecturers of IF3280 Socio-informatika dan Prefosionalisme, Dr. Ir. Rinaldi Munir, MT., Dr. Eng. Ayu Purwarianti, ST., MT., and Dessi Puji Lestari ST., M.Eng., Ph.D. Thanks to the Author's mother who always supporting the Author to focus on his study. Thanks to Unit Robotika ITB, especially Wheeled Soccer Robot Team of ITB a.k.a DAGOZILLA ITB Team for allowed me to share this project in this paper. Last but not least, thank you to all of the people around the Author who have accompanied the Author through his hardest times.

References

[1] Bradski, G., & Kaehler, A. (2008). Image Processing, Learning OpenCV (pp. 109-141). Sebastopol, CA: O'Reilly Media Inc.

[2] Bradski, G., & Kaehler, A. (2008). Image Transforms, Learning OpenCV (pp. 144-162). Sebastopol, CA: O'Reilly Media Inc.

Biography

Muhammad Kamal Nadjieb is a Computer Science/Informatics student at Institut Teknologi Bandung. Now, he is on a project to win KRI (Kontes Robot Indonesia) in Wheeled Soccer Robot Competition.

Bandung, 5 May 2017.

[illegible]

Muhammad Kamal Nadjieb