Object detection using Hough transform

Project Report

Group 765 / ROB7

Aalborg University
Electronics and IT

AALBORG UNIVERSITY
STUDENT REPORT

Electronics and IT
Aalborg University
http://www.aau.dk

Title:
Circle detection using Hough transform

Theme:
Advanced Mobile Robotics

Project Period:
Fall Semester 2021

Project Group:
Group 765

Participant(s):
Daniel Moreno París
Lucas Louis René Sebille
Reshad Zadran
Sabrina Kern

Supervisor(s):
Chris Holmberg Bahnsen

Copies: 1

Page Numbers: 15

Date of Completion:
November 5, 2021

Abstract:

As Object detection in images is an ongoing challenge in computer vision, Mohamed Rizon et al. [1] propose an algorithm combining a separability filter and Circular Hough Transform as one solution for identifying objects. The objects that they want to detect are coconuts, but they suggest the algorithm as suitable for all circular objects. The aim of this project is to reproduce the algorithm and to test whether it works as promised. It could be shown that the algorithm is reproducible and that it works using images with single circle formed objects. However, when it was tested with more complex images with multiple objects or noise, it fails.

Contents

Chapter 1

Introduction

As Object detection in images is an ongoing challenge in computer vision, Mohamed Rizon et al. [1] propose an algorithm combining a separability filter and Circular Hough Transform as one solution for identifying objects. The objects that they want to detect are coconuts, but they suggest the algorithm as suitable for all objects with circular shape.
The aim of this project is to reproduce the algorithm and to test whether it works as promised. The chapters 2, 3 and 4 are giving a general introduction into the preprocessing steps, the separability filter and the Hough Transform. The implementation and the results are explained in the chapter 5. Finally, the conclusion is presented in chapter 6.

Chapter 2

Preprocessing

Four steps are performed to prepare the image for further processing. The first step is gray-scaling, where the RGB image is transformed to an image in the shades of gray. This step is necessary so that the image can be manipulated further on. Thereupon, the method of Histogram Equalization is applied in order to get an higher contrast in the image. This facilitates the process of finding edges in the next step.
A histogram represents the intensity distribution of pixels of each lightness value in the image. The method of Histogram Equalization spreads out the most frequent intensity values. This will ideally lead to an increased contrast in the image, as areas with a low local contrast will gain a higher contrast. However, this method can also cause disadvantages as a successful result depends on the input image. Furthermore, the contrast of background noise could also be amplified and therefore more visible. [2] The figure 2.1 shows how the method works. The original histogram of an image is displayed on the left side, with a high amount of pixels clustered in the middle. The histogram on the right side shows the result after the Histogram Equalization.

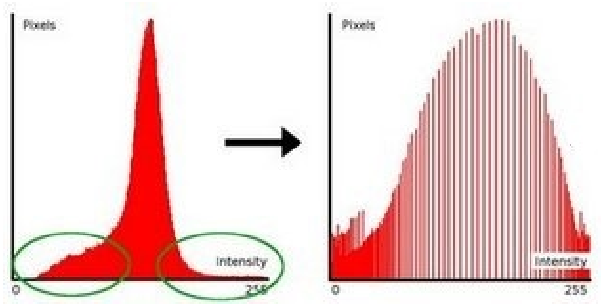


Figure 2.1: Histogram Equalization [3]

Afterwards Canny edge detection is used for segmentation of the image. This method is a popular edge detection method that follows the idea that edges can be detected through a change of the pixels intensity. It consists of four stages. The first one is to remove the noise in the image with the help of a Gaussian filter. Afterwards, the gradient of each pixel is calculated in horizontal and vertical direction based on the grey-level. [4] In order to achieve thin edges in the image, the next step is to perform non-maximum suppression to remove unnecessary pixels. Therefore every pixel is checked if it is a local maximum compared to its two neighbors concerning the gradient direction. If not, the pixel is suppressed. As the result of this is a binary image with pixel-thin edges but still with variation regarding the edges' intensity, another processing step is needed. The concept of Hysteresis Thresholding decides which pixels should be finally taken as edges. Therefore two threshold values are defined, a minimum and a maximum value. All pixels above the maximum threshold are considered as strong pixels and are valid edges. All pixels under the minimum value are discarded. Pixels in between the thresholds are only taken as edges, if they are connected with a group of pixels which contains at least on pixel above the maximum threshold. This finally results in strong edges. [5]
The last step of preprocessing are Morphological Operations. Those are operations are used to remove noise, isolate individual elements or to find holes in an image. An closing morphology is employed to close small gaps and removes noise. An opening morphology is then used to open up spaces and remove unimportant information. [1]

Chapter 3

Separability filter

3.1 Circle template

Once the preprocessing is completed, it is now possible to apply the separability filter. The separability filter used here was proposed by Fukui and Yamaguchi in a previous paper [6] where the objective was to perform facial recognition by extracting facial features like pupils, nostrils and mouth edge using a separability filter. This filter is based on a template method, which means that it uses a particular template to detect the best object candidates. In the case of this paper, a circular template is used like shown in figure 3.1.

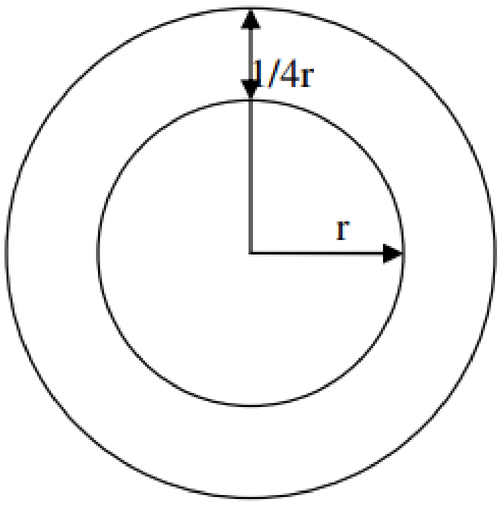


Figure 3.1: The circle template to detect the object candidate

The separability filter is used in the system proposed by the paper in order to detect the best objects candidates that we are looking for (in this case coconuts).

Here the purpose of the filter is to obtain the possible coordinates of each coconut that is present in a picture. After that, another technique will be used to detect the presence of circular shapes.
Next, we will present the different parts that make up this filter and explain how it works.

$$A = \sum_{i=1}^{N} (I(x_i, y_i) - P_m)^2 \tag{3.1}$$

$$B = n_1(P_1 - P_m)^2 + n_2(P_2 - P_m)^2 \tag{3.2}$$

$$S = B/A \tag{3.3}$$

Where:

$n_k(k = 1, 2)$ is the number of pixels inside the region $R_k(k = 1, 2)$
$N = n_1 + n_2$ is the number of pixels inside both regions
$P_k(k = 1, 2)$ is the average intensity inside the region $R_k(k = 1, 2)$
P_m is the average intensity inside the union of the regions R_1 and R_2
$I(x_i, y_i)$ is the intensity values of the pixels (x_i, y_i) in the regions R_1 and R_2
S is the separability between region R_1 and region R_2

In order to detect the best object candidates in the image, the proposed algorithm starts by placing the template of figure 3.1 in the image. Then, the center of the template is moved from a certain amount of pixel from left to right and from top to down as shown in figure 4.1. Moreover, the radius r of the circle is changing within an interval $[r_l, r_u]$, with r_l being the lower bound and r_u the upper bound. By performing the division of the equation **(3.2)** and **(3.1)**, the algorithm can provide the separability S between the two regions R_1 and R_2 in the template with size r.
After this step, the separabilities are compared for each pixel (x_i, y_i) and each radius (r), we can use the notation $S(x_i, y_i, r)$. The algorithm will then select the triplets (x, y, r) that provide the local maxima of the separability. The circles obtained by those triplets of values are selected as object candidates and are the most likely to contain coconuts.

Chapter 4

Hough Transform

4.1 Circular Hough Transform

The Hough Transform (HT) is an algorithm that was patented in 1962 by Paul V. C. Hough whose original purpose was to identify complex lines in photographs. Since then, the algorithm has been modified and improved to be able to recognise other different shapes such as circles.
The Circular Hough transform (CHT) is one of the several applications of the Hough transform (HT) with the purpose of finding circular patterns within a picture. It was employed for the first time by Duda et al. [7] This method can recognise circles even if there is noise in the image.
In order to define the Hough transform for the circle, the first thing to do is to describe the circle pattern by the equation (3.1) where (x_0,y_0) are the coordinates of the centre of the circle and r is the radius.

$$(x_p - x_o)^2 + (y_p - y_o)^2 = r^2 \tag{4.1}$$

To find circles using the HT, an accumulator array with three dimensions is necessary.
Then each point in the image votes for the circles on which it could be, therefore, the array elements with the highest number of votes indicate the existence of a shape. Once this procedure is finished, the peaks in the accumulator are searched for and the radius and the centre of the circle are obtained. If the radius were known beforehand, only a two-dimensional accumulator would be needed.
Many modifications of CHT are available to reduce the computational complexity or increase the detection rate. For the detection of circular shaped objects, the edge orientation information is used to increase the performance of CHT. The use of edge orientation information restricts the possible centre positions of each edge point. Therefore, applying this method, it is only necessary to draw an arc perpendicular to the edge orientation at a distance R from the edge point.

4.2 Triplets from separability filter

Once the segmentation steps have been completed, the separability filter technique and the circular HT process are applied to the image.
As the separation filter has been employed previously, when using the CHT, the radius r is known in advance because the shape of the object is a peculiar characteristic of each one, so the radius r can be treated as a known parameter.

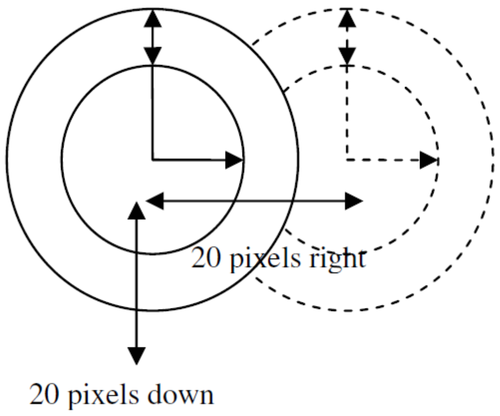


Figure 4.1: The template moves 20 pixels from left to right and then 20 pixels down, starting from the left again.

Chapter 5

Implementation and Results

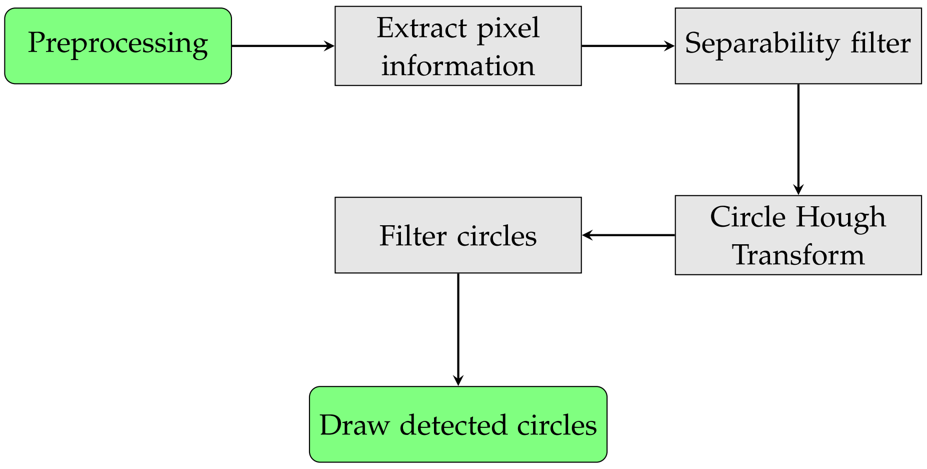


Figure 5.1: Flowchart of circle detection process

In the flowchart 5.1 the overall coding process of the implementation is outlined. The implementation is split into six steps. Each step is defined in a function of its own and called in the main function when an image with circular objects needs to be detected. The function and implementation of each step will be explained in this chapter.

5.1 Preprocessing and pixel information extraction

The first steps of the implementation is to preprocess and extract the pixel information from the image. The preprocessing is done by opencv functions for smoothing, gray-scaling and edge detection.
The listing 5.1 shows how the pixel information is obtained. A circular mask at the circle center with a radius is defined. All the pixels within the radius of the mask

are set to true and all the other pixels to false. All the entries of the mask are looped through. If the entry is true, the pixel is counted within the radius of the circle so the total amount of the variable (pixsTotal) is incremented. Furthermore the index which returns true is used on the image (img[y,x].item()) to get the intensity value at that pixel position and their summed up to get the total intensity in the circle. A similar approach is used to get the pixel information in the union of R1 and R2. The pixel information is needed to calculate the separability.

```python
def pixiExt(img,x0,y0,radius):
    ...
    dist_from_center = np.sqrt((x0-x)**2 + (y0-y)**2)
    mask0 = dist_from_center <= radius
    mask1 = dist_from_center <= radius*1.25
    mask2 = mask1 & ~mask0

    height1,width1 = mask0.shape
    for y, x in itertools.product(range(0,height1), range(0,width1)):
        if mask0[y,x].any() == True and x>=0 and x<=width and y>=0 and
     y<=height:
            totalInten += img[y,x].item()
            pixsTotal += 1
            pixsLoca.append((img[y,x].item()))
        if mask2[y,x].any() == True and x>=0 and x<=width and y>=0 and
     y<=height:
                unionPix += 1
                unionInte += img[y,x].item()
                uPixVal.append((img[y,x].item()))
    return pixsTotal,totalInten,uPixVal,unionPix,unionInte
```

Listing 5.1: Pixel information extraction

5.2 Separability filter calculation and CHT

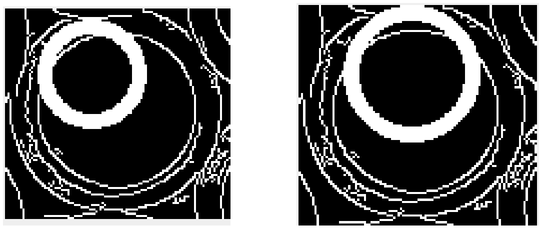
The separability is calculated as it is explained in chapter 3 and the code implementation can be seen in the listing 5.2.

```python
def paraCal(pR1,pR2,inR1,inR2,inValR1,inValR2,uPi,uIn): #(p = pixels ,
    in = intensity,upi = pixels in union, uIn = intensity of union)
    ...
    if uIn and uPi != 0:
        uAvgIn = uIn/uPi #average intensity of union
        for i in range(len(inValR1)):
            Areg += ((inValR1[i]-uAvgIn))**2 #The average intensity of
     region R1
            #print(len(inValR1))

        for i in range(len(inValR2)):
            Areg += ((inValR2[i]-uAvgIn))**2 #The average intensity of
     region R1
```

```
            #print(len(inValR1))

        Breg = (pR2*(((inR2/pR2)-uAvgIn)**2))+(pR1*(((inR1/pR1)-uAvgIn
    )**2))
        if Areg and Breg !=0:
            regRatio = Breg/Areg

            return regRatio
```

Listing 5.2: Separability calculation

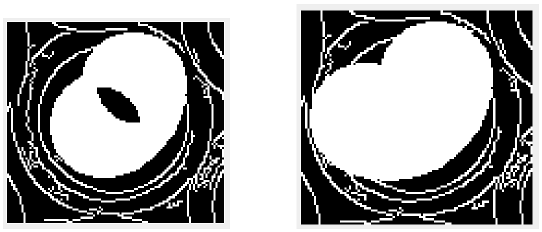
The separability is calculated for every pixel in steps of 20 pixels for the whole image. If the separability is higher then the threshold of 0.1, the circle centers coordinates and radius are saved in a list. The figure 5.2 shows regions with a detected separability over 0.1.



(a) Union of R1 and R2 **(b)** Union of R1 and R2

Figure 5.2: Separability of 2 regions where its over 0.1

For the region with a separability over 0.1, their triplet (x,y,r) value is fed into the CHT. The CHT votes for the most optimal center with the radius that have been determined by the separability filter. From the list, the five highest voted centers are chosen to be processed further. The figure 5.3 shows the visualised CHT voting.

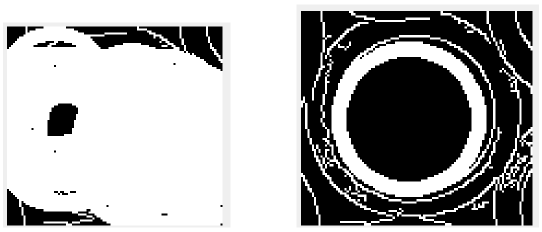


(a) Circle Hough Transform **(b)** Circle Hough Transform

Figure 5.3: Voting process of the Circle Hough Transform

5.3 Circle filtering

It is necessary to filter circles in order to avoid the image getting polluted with redundant circles. Figure 5.4 shows all the circles in one image and a image where the circles have been reduced down to one.



(a) All circles **(b)** Filtered circles

Figure 5.4: Filtering of circles process

The filtering is done by looking at the euclidean distance between the detected circles. If the distance is lower then the given threshold, the circles will get merged. The x and y coordinates gets averaged and the biggest radius is chosen. Circles that stands alone with no neighbors within the given threshold are filtered out. The recommended threshold to use for filtering circles out is a bit below the diameter of the maximum radius that a circle can have. The listing 5.3 shows the code of what have been explained.

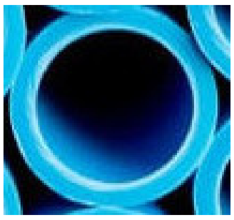
```python
while circleInRang:
        ...
        dist = np.linalg.norm(np.array((int(sortArr[ii][0][0][0]),int(sortArr[ii][0][0][1])))-np.array((int(sortArr[jj][0][0][0]),int(sortArr[jj][0][0][1]))))
        if dist <= Rmax*1.7 and ii != jj:
            jCicle.append(sortArr[jj])
            ...
        ...
            if len(jCicle) != 0:
                for i in range(len(jCicle)):
                    #print("range af circle",len(jCicle))
                    avgX += int(jCicle[i][0][0][0])
                    avgY += int(jCicle[i][0][0][1])
                    maxR = int(jCicle[i][0][0][2])

                    if maxR >= rdiMax:
                        rdiMax = maxR
                ...
            ...
    ...
```

Listing 5.3: Circle filtering

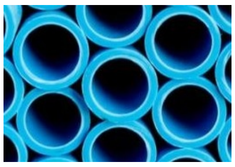
5.4 Testing on ten different images

5.4.1 Images to be processed

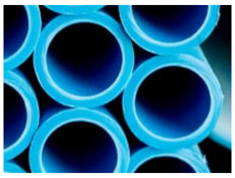
The following images were used for testing.


(a)

(b)

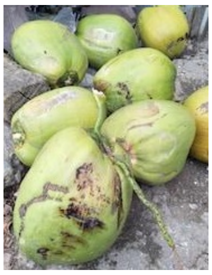

(c)

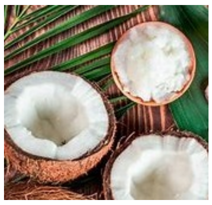
(d)


(e)

(f)

(g)


(h)


(i)

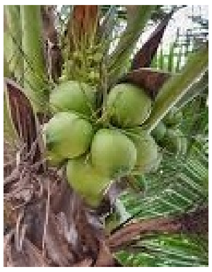

(j)

Figure 5.5: Original cropped images

5.4.2 Processed images

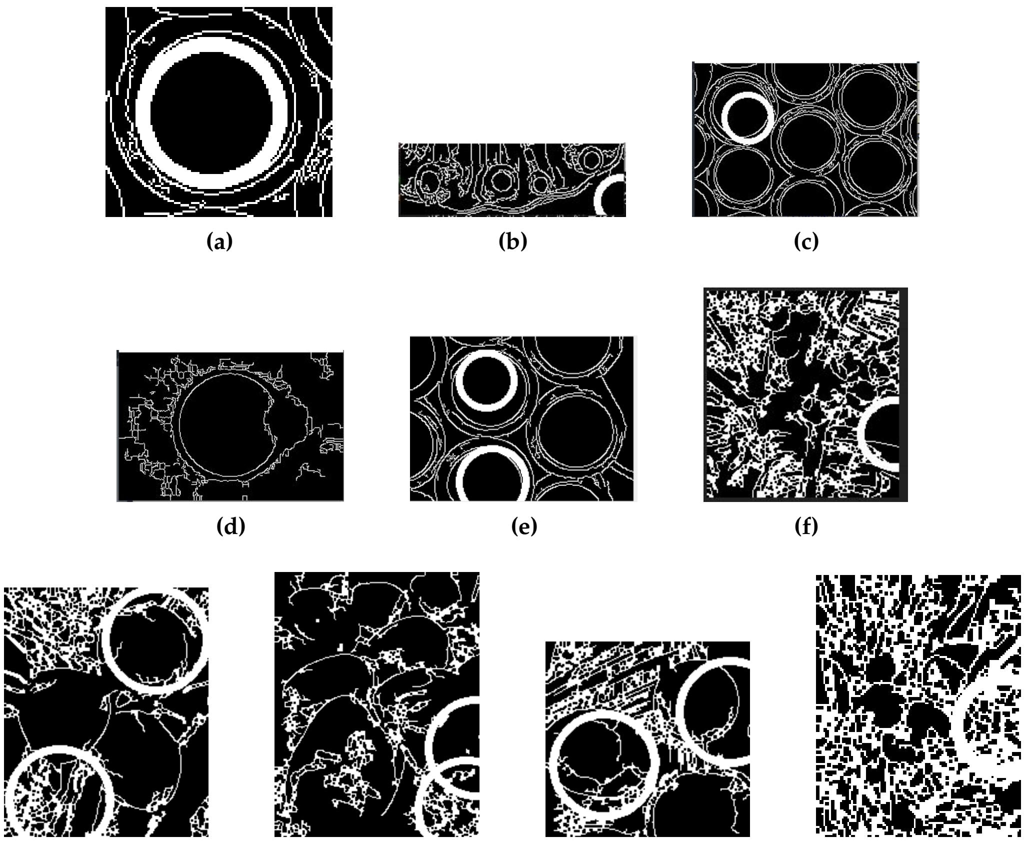


Figure 5.6: Processed images

The figure 5.5 shows the different images which need to be processed and the figure 5.6 shows the results. The images show that the algorithm does well for single clear circular images like in figure 5.6 (a). When there is more noise, edges and objects in the images, the algorithm starts to fail. In the figure 5.6 (d) the algorithm does not even manage to find any circles at all. There are a lot of different variables to play with that makes the algorithm fail. These are for example the step size, too many edges, not enough edges and static thresholds. With a step size of 20 pixels as proposed in the paper, good centers can be skipped. If there are few or too many edges in the region, the separability filter with its threshold will not count good centers. To get better results, the threshold of merging circles should not be dependent on the maximum radius but on the detected circles radius.

Chapter 6

Conclusion

The goal of this project was to reproduce the method used in [1]. With the clarification of the mathematics by [8] for the separability filter, the goal was reached. The prepossessing step was achieved by using the build-in functions of OpenCV for gray scaling, histogram equalization, canny edge detection, closing and opening morphology. The variables of the functions were changed in such that the processed image was a suitable representation of the input image. After prepossessing, the separability filter was used on the image to find good circular regions. The output of the separability filter was the center and radius of a circle. This information was fed into the HT to vote for the best center. Then all the circles which were redundant were removed by the circle filtering step, which lead to the final result.
The final product was able to detect single circle formed objects, but when more complex images with multiple objects were tested, the algorithm failed. However, the algorithm could be further developed and tested using more time to adjust the different settings and values of the steps and the variables.

Bibliography

[1] Mohamed Rizon et al. “Object Detection using Circular Hough Transform”. In: *American Journal of Applied Sciences* 2.12 (2005), pp. 1606–1609.

[2] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[3] *Histogram Equalization*. `https://docs.opencv.org/3.4.15/d4/d1b/tutorial_histogram_equalization.html`. 2021.

[4] Andreas Møgelmose. *Sobel Edge Detection*. Aalborg University, Course: Advanced Robotic Perception (ROB7). 2021.

[5] Andreas Møgelmose. *Canny Edge Detection*. Aalborg University, Course: Advanced Robotic Perception (ROB7). 2021.

[6] Kazuhiro Fukui and Osamu Yamaguchi. “Facial feature point extraction method based on combination of shape extraction and pattern matching”. In: *Systems and computers in Japan* 29.6 (1998), pp. 49–58.

[7] R.O. Duda and P.E Hart. *Use of the Hough transformation to detect lines and curves in picture.* 1972.

[8] Tsuyoshi Kawaguchi, Mohamed Rizon, and Daisuke Hidaka. “Detection of eyes from human faces by Hough transform and separability filter”. In: *Electronics and Communications in Japan (Part II: Electronics)* 88.5 (2005), pp. 29–39.