UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Computer Vision and Image Processing

Project Report

Students:
Andrea Dario
Matteo Caselli
Matteo Boldini

Academic year 2023/2024

Contents

Introduction

Project presentation

This document shows how we developed a software solving two different tasks in locating the possible defects in a cap composed by a white plastic shell and a silver cardboard liner.

Figure 1: Image of a cap, shell is 1, liner is 2

First Task

The first task is to perform the segmentation of the cap mouth, outlining it, and to find the presence of possible defects. The features to be printed are the following:

1. position of the cap center
2. diameter of the cap mouth
3. liner presence
4. liner quality (if the liner is present, is it complete?)

In case of incomplete liner the software also highlights its straight edge.

Second Task

The second task is to outline the liner and to print the following features:

1. position of the liner center
2. diameter of the liner

Software presentation

We will now break down the software and explain how we managed to complete the assigned tasks and the reasons behind the utilised functions. In the first lines of code some values are set, these values allow us to access the images and some of them are set by trial and error (like the thresholds) and will be useful later on.

First Task

The core of the software is the **for** loop elaborating all images one by one, inside this loop the following operations are carried out to tackle the first assignment:

- image acquisition and filtering
- cap mouth outlining
- cap analysis
- liner's straight edge outlining (if incomplete)

Image acquisition and filtering

Each image is read through the method **cv2.imread()** and converted from coloured to grayscale using **cv2.cvtColor()**, this elaboration is performed because we will need to work with grayscale levels in order to fulfill the tasks. Each image is put in the list **images**. A filtering operation is carried out through the **cv2.GaussianBlur()** command, this applies a gaussian filter with 3x3 kernel ans sigma automatically selected and its aim is to reduce noise and false circle detection.

Cap mouth outlining

To outline the cap mouth we use the Hough transform from cv2 library, the function is **cv2.HoughCircles()**, it takes the filtered image computed in the precedent stage as input and gives back a 3 element vector containing the **position of the center (x and y coordinates)** and the **radius** of the

cap. We now need to draw the found circle, using the function **cv2.circle()** inside a for loop we are able to accomplish this task. The cited function takes as input the image, the center coordinates and the radius of the circle to draw (along with the desired color and thickness). Since we want to outline both the center of the cap and the cap mouth we call the function two times, the only difference between the calls is that the radius is put equal to one to draw the center and takes the value computed before to draw the cap mouth.

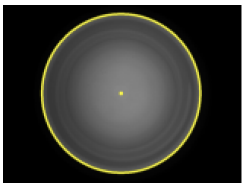


Figure 2: Cap with outlined mouth and center

Cap analysis

In the same loop used to draw the circle we save the position of its center and the value of the diameter, computed doubling the radius. To answer to the question "Is the liner missing?" we know we can use the mean value of the images and threshold them (since the caps without liner have higher mean). We generate a circular mask with the same dimension of the liner and then we compute the mean value of the image overlaying the mask over the image, in this way all the pixels "outside" the mask are not included in the mean computation (differences in background lighting conditions don't "falsify" the mean). The mean value is saved in the **liner** variable, that is compared to the threshold, obtaining the answer to the given question, saved in **linerPresent** variable.

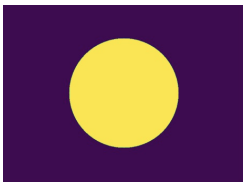


Figure 3: Image of the mask of one cap

To answer the question "Is the liner incomplete?" we applied the Canny edge detector taken from the cv2 library **cv2.Canny()** giving as input the grayscale image in which all the pixels outside the mask are put to zero. To obtain a better result we also applied a dilation after the edge detector, using the function **cv2.dilate()** and a 3x3 kernel, this operation increases

the area of the features found with canny edge detector, giving us stronger edges. Lines are drawn using the **cv2.HoughLinesP()**, function that finds line segments in a binary image using the probabilistic Hough transform, the input is the dilated image and the output is saved in the variable **lines**. The answer to the question is obtained checking the value of **lines**, if it is bigger than zero, then the liner is incomplete (the result is saved in the variable **defected**). The cap analysis is then printed, showing all the requested characteristics.

```
image: d_20 [ mean = 0.33 --> Liner is incomplete ]
cap center = (384, 290), cap diameter = 508.00
```

Figure 4: Results obtained from one cap analysis

Liner's straight edge outlining (if incomplete)

Using a if statement we check the value of **defected**, if the value is true we draw the line highlighting the straight edge of the liner using the function **cv2.line**.

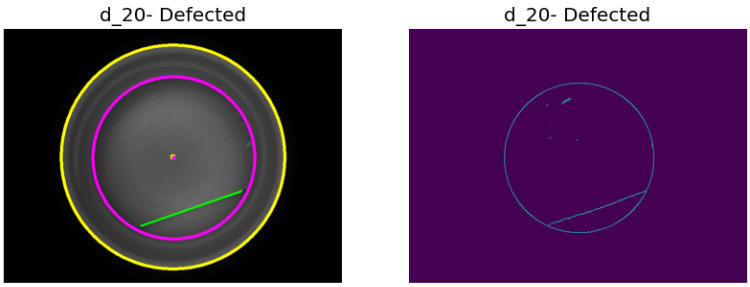


Figure 5: Liner straight edge is outlined, found edge is shown nearby

Second Task

The second task is pretty similar to the first part of the first task. To find the liner border we once again used the Hough transform **cv2.HoughCircles()**, this time with lower parameters and putting a maximum radius value to avoid finding the cap mouth border. The found circle and its center are then drawn using the function **cv2.circle()**, using the center coordinates with radius equal to one to highlight the center and the radius of the found circle to outline the liner border.

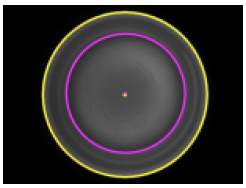


Figure 6: Liner and its center are outlined

The liner center and diameter are then saved in the apposite variables **linerCenter** and **linerDiameter**. The liner values are printed in the cap analysis, after the cap values.

```
--------------------------------------------------------------------------------
image: d_21 [ mean = 0.33 --> Cap is not good but liner is present ]
cap center = (388, 286), cap diameter = 508.00
liner center = (384, 286), liner diameter = 370
```

Figure 7: Results obtained from one cap full analysis

Results

Verifying the results obtained one by one we can state that the software performs well (all the analysis are correct). The defective caps are all identified and the particular defect is shown in the log and in the image (if necessary). The following images show the full analysis of all caps and the outlined segments in the ones with incomplete liner.

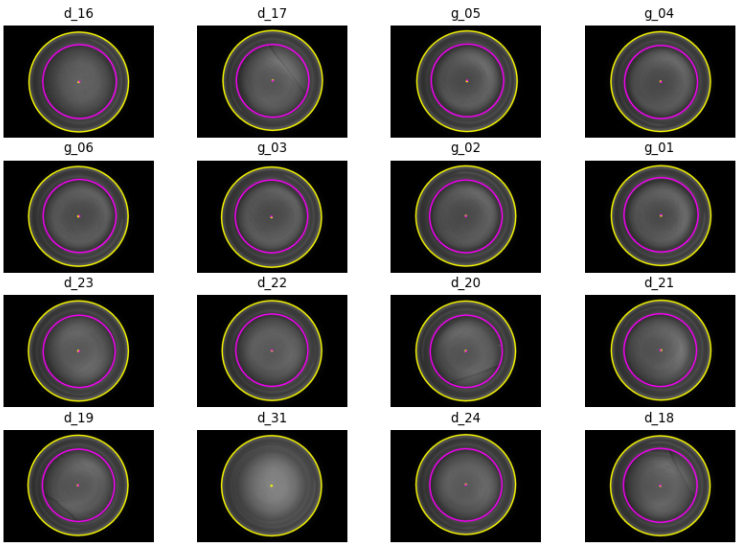


Figure 8: Mouth and liner detection in all caps

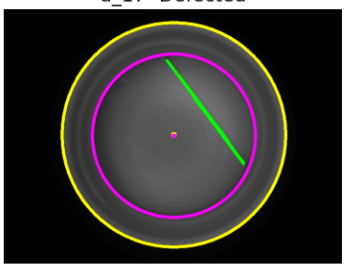
```
image: d_16 [ mean = 0.31 --> Cap is not good but liner is present ]
cap center = (384, 292), cap diameter = 508.00
liner center = (388, 290), liner diameter = 376
image: d_17 [ mean = 0.34 --> Liner is incomplete ]
cap center = (386, 284), cap diameter = 508.00
liner center = (386, 286), liner diameter = 370
image: g_05 [ mean = 0.32 --> Cap is good and liner is present ]
cap center = (390, 288), cap diameter = 512.00
liner center = (392, 284), liner diameter = 370
image: g_04 [ mean = 0.32 --> Cap is good and liner is present ]
cap center = (386, 290), cap diameter = 508.00
liner center = (388, 292), liner diameter = 372
image: g_06 [ mean = 0.32 --> Cap is good and liner is present ]
cap center = (382, 288), cap diameter = 508.00
liner center = (388, 286), liner diameter = 372
image: g_03 [ mean = 0.32 --> Cap is good and liner is present ]
cap center = (380, 292), cap diameter = 510.00
liner center = (380, 288), liner diameter = 370
image: g_02 [ mean = 0.32 --> Cap is good and liner is present ]
cap center = (384, 286), cap diameter = 510.00
liner center = (384, 288), liner diameter = 370
image: g_01 [ mean = 0.32 --> Cap is good and liner is present ]
cap center = (388, 284), cap diameter = 508.00
liner center = (388, 280), liner diameter = 378
```

Figure 9: First 8 caps analysis

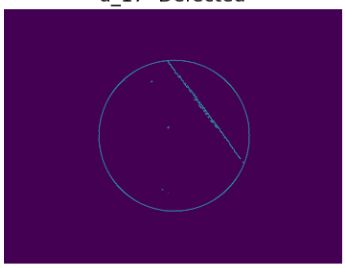
```
image: d_23 [ mean = 0.33 --> Cap is not good but liner is present ]
cap center = (382, 290), cap diameter = 510.00
liner center = (386, 292), liner diameter = 368
image: d_22 [ mean = 0.33 --> Cap is not good but liner is present ]
cap center = (382, 288), cap diameter = 508.00
liner center = (382, 286), liner diameter = 368
image: d_20 [ mean = 0.33 --> Liner is incomplete ]
cap center = (384, 290), cap diameter = 508.00
liner center = (386, 292), liner diameter = 368
image: d_21 [ mean = 0.33 --> Cap is not good but liner is present ]
cap center = (388, 286), cap diameter = 508.00
liner center = (384, 286), liner diameter = 370
image: d_19 [ mean = 0.33 --> Liner is incomplete ]
cap center = (380, 286), cap diameter = 510.00
liner center = (382, 286), liner diameter = 368

image: d_31, [ mean = 0.42 --> NO LINER! ]
cap center = (380, 288), cap diameter = 510.00

image: d_24 [ mean = 0.34 --> Cap is not good but liner is present ]
cap center = (384, 282), cap diameter = 508.00
liner center = (384, 284), liner diameter = 368
image: d_18 [ mean = 0.33 --> Liner is incomplete ]
cap center = (384, 288), cap diameter = 508.00
liner center = (384, 286), liner diameter = 376
```

Figure 10: Last 8 caps analysis

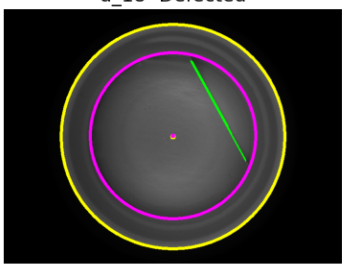
d_17- Defected



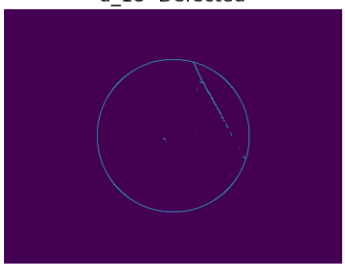
d_17- Defected



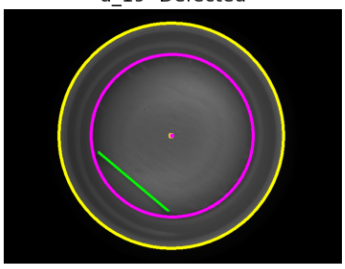
d_18- Defected



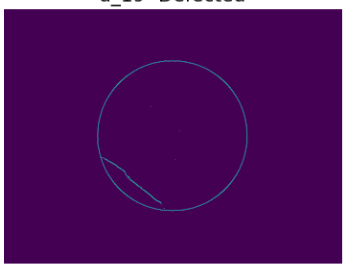
d_18- Defected

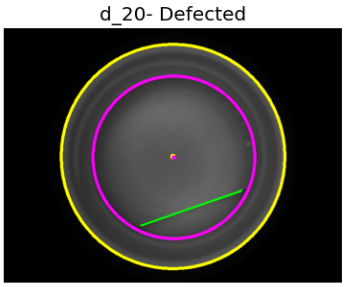


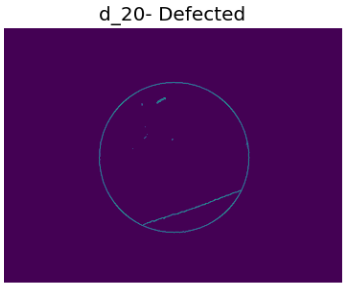
d_19- Defected

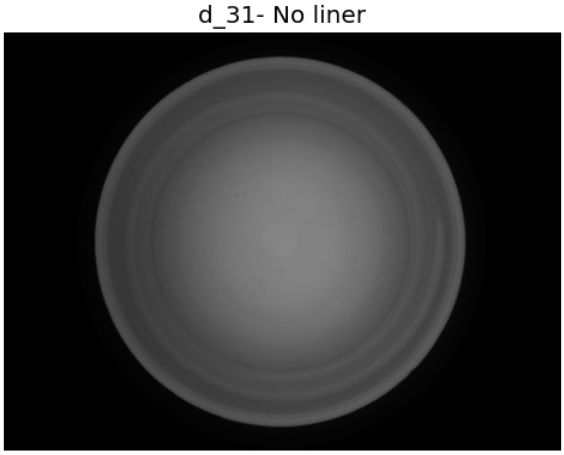


d_19- Defected




d_20- Defected


d_20- Defected


d_31- No liner