UNIVERSITY OF BURGUNDY

MASTER IN MEDICAL IMAGING AND APPLICATIONS

Image Processing
Soft Drink Bottling Plant

Submitted To:
Dr. Désiré Sidibé

Submitted By :
Prem Prasad
Manju Kumar Basavaraj
Oliver Sehou

January 13, 2019

Contents

1 Introduction

In large production facilities, a key step is the identification of faults in products that leave the facility. This step is crucial as it allows only well manufactured products to leave the facility while the damaged products are removed and recycled.

The bottling company requires a vision system to automatically identify a number of different faults that may occur during the filling, labelling, and capping stages in the production line, so that they can be identified before they are sent to packaging.

In this project, we are creating an algorithm in MATLAB that detects faults in bottles that leave the bottling production line.

2 Task Specification

We are given images that contain three bottles per image, but we are only taking into consideration the **bottle in the center** as the ones on the sides have their own images with them as the bottle under consideration (i.e. center). Thus images with faulty side bottles are ignored by our system.

These are the tasks that need to be done by our visual inspection system:

1. bottle **under-filled** or not filled at all
2. bottle **over-filled**
3. bottle has **label missing**
4. bottle has label but **label printing has failed** (i.e. label is white)
5. bottle **label is not straight**
6. bottle **cap is missing**
7. bottle is **deformed** in some way

3 Image Processing Techniques Used

The following image processing techniques have been used in our visual system for identifying the faults in the images.

3.1 Image Cropping

This is an image processing tool that allows us to consider only a portion of the original image. This is useful as it allows us to segment a portion of the original image from the

rest.

For example, to identify if a bottle cap is missing, we have cropped the image to contain the approximate region where the bottle cap will be present. Once we have this region, we can apply other image processing techniques to conclude if the cap is present or not.

Another example is for finding if the bottle has a label or not. We crop the section of the original image that contains the label, and then apply other techniques to conclude is we have a missing label or not.

Note: we have taken advantage of the fact that all images have the bottles in approximately the same position. If this were not the case, locating the bottle for every image requires other techniques. But for our case, the center bottle is in relatively the same position with only a small degree of shifting in either direction. To handle the shift in either direction, we have taken a slightly larger crop window.

We have created a function that when given an image I, an x,y coordinate, and length of crop in in either direction $length_x$ and $length_y$, we return an image with $length_x$ and $length_y$ dimension having the region of cropped image from the original image specified by x,y.

3.2 Thresholding

This is a very simple and effective image processing technique that allows us to replace a pixel value to either black or white depending on if its intensity level $I_{i,j}$ is less than or greater than a *threshold intensity value* T respectively. That is:

$$I_{i,j} = 0; \qquad I_{i,j} < T$$

$$I_{i,j} = 255; \qquad I_{i,j} >= T$$

For thresholding to be effective, selecting the right threshold value is very important. In our project, since most images have similar lighting condition, similar colors and similar intensity values throughout, we have use static threshold values to segment the images of interest.

3.3 Summation of Intensity Levels

This may seem like a very simple technique, but when used correctly, we can make important decisions with this information. This basically sums all the intensity levels of an image and gives a total sum. We have made use of this after a cropped region of interest has been created.

Following are the steps taken to conclude to help make decisions:

1. crop the original image to return the section that contains the label of the bottle of interest
2. find the sum of the intensity levels in the cropped image.
3. if the sum is greater than a specific value, we conclude with one result, while if not, we conclude the other.

3.3.1 Examples of Use:

(a) **Label Print Missing**

If a label has no print, then the label will contain high number of white pixels (which are high intensity pixels) while the bottles having label print will have large number of lower intensity values, thus reducing the overall sum of the intensity levels in the image.

(b) **Bottle Cap Missing**

The region of interest to find this fault is the region containing the bottle cap. If a bottle has a cap, it will have a larger number of darker colored pixels due to presence of the cap (in gray-scale), thus reducing the sum of the region of interest. If the bottle has a missing cap, it will have more higher pixel intensity values due to the missing cap, thus increasing the overall sum of the cropped image of interest.

To find the sum that differentiates one of 'cap missing' and 'cap present', we find the sum of intensity levels of the bottles having a cap, and the sum of the bottles missing a cap, and find a value in the middle that is use to differentiate the two. If the sum of the intensities is less than that of the newly found differentiated value, we conclude that it has a bottle cap, and if greater, has a missing bottle cap.

3.4 Histograms

Histograms are a representation of the intensity distribution of pixel values in an image. This is a useful technique that gives information about the frequency of occurrences of a pixel intensity level in an image.

Histograms of an image can have a *bin* size specified that provides the range of values that fall into that particular bin. For an image of 255 intensity levels, we can have a maximum of 255 bins, one for each intensity level.

4 Flow Chart

Following figure 1 is a simple flowchart representing the work-flow of our project. We first find if the bottle has a 'missing cap'. Then we calculate if the bottle is 'over-filled', 'under-filled', or 'normally filled'. Next, we find if the bottle has a 'label' or not. If the bottle is found to have a missing label, we go straight to finding if the bottle is 'deformed'. This is because finding the other two faults of 'successful label print' and 'label straight' is useless. If a label is detected, then we proceed to find if the 'label print' has been successful or not. And later we find if the 'label is straight'. Finally, we check if the bottle is deformed

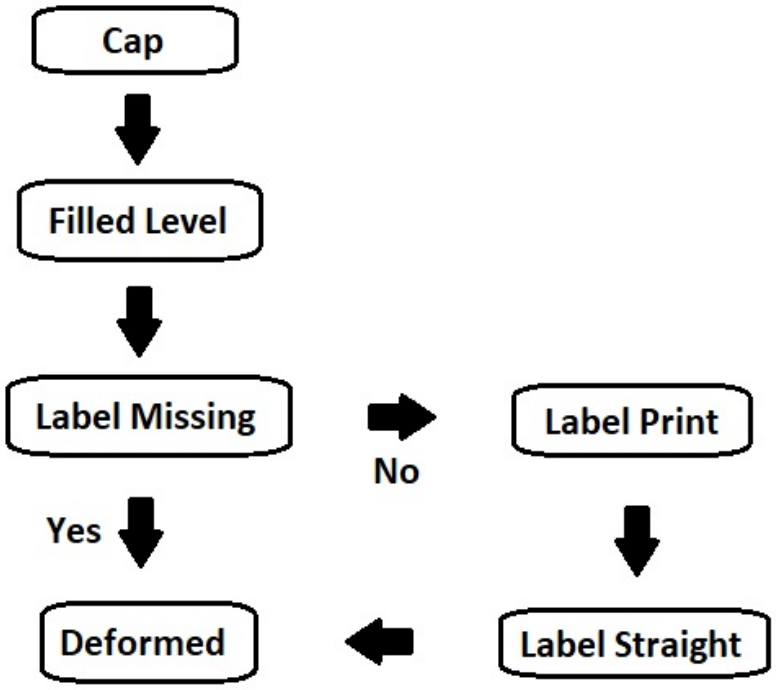


Figure 1: Work flow

5 Fault Finding Steps and Reasoning

5.1 Cap Missing

- **Algorithm**

(a) Cap Present (b) Cap Missing

Figure 2: Window of Interest

1. converting RGB image to gray-scale
2. crop out the region around the bottle cap shown in fig 2a and 2b
3. find the sum of the intensity levels in the cropped image

$$\sum I = \sum_{i=1}^{m} \sum_{j=1}^{n} I_{i,j}$$

where m, n are number of rows and columns

4. Condition:

$$Result = \begin{cases} 1, & \text{if } \sum I < 1,175,000. \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where:

(a) 1: Bottle Cap Missing

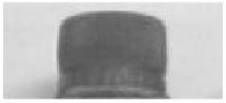
(b) 0: Bottle Cap Present

- **Reasoning**
When we find the sum of the intensity levels of the cropped region, we find the following:

– **for missing bottle cap:** on average the sum is approximately 1,300,000 (due to less dark regions)

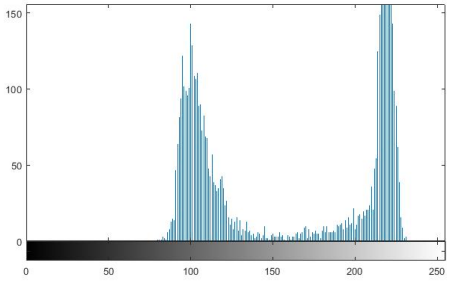
– **for bottle cap present:** on average the sum is approximately 1,050,000.

The following figure 3 shows the cropped region of interest along with its histogram. We see that in 3c, the number of pixels in the dark region is more than 3d. This is due to the fact that the *cap present* brings in a high frequency of low pixel intensity values as seen in 3a, while 3b has a low frequency of low intensity pixels.

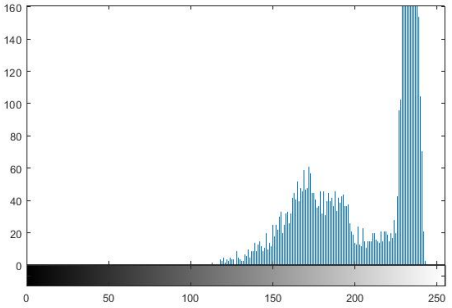


(a) Cropped Cap Present

(b) Cap Missing



(c) Network 3



(d) Network 4

Figure 3: Window and its Histogram

So, to differentiate, we split the difference to find the sum that best separates the missing bottle caps from the non missing bottle caps. We find that 1,175,000 is that sum.

- **Results**
  Figure 4 is a graph that shows the above information for images for the *missing cap* and *normal* images:

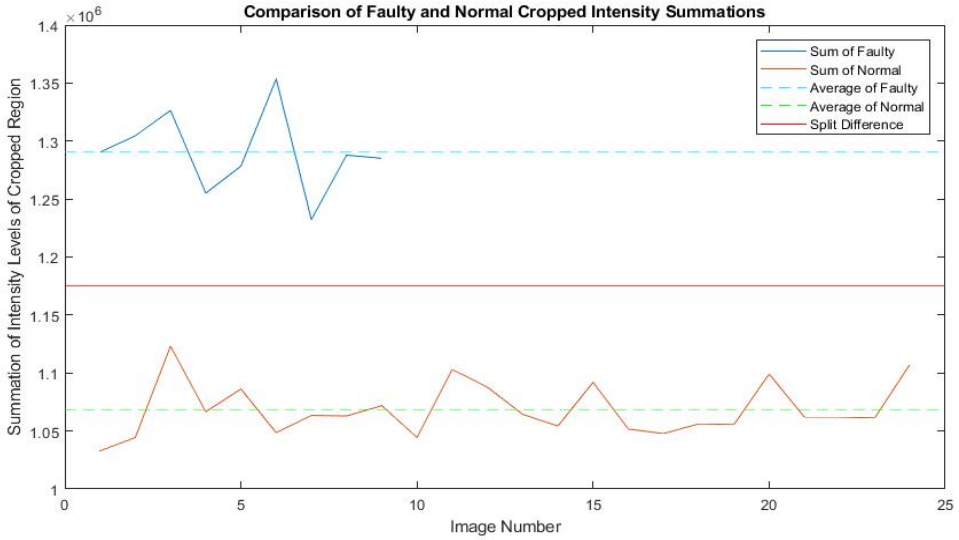


Figure 4: Comparison of Faulty and Normal Cropped Intensity Summations

5.2 Filled Level

- **Algorithm**

(a) Overfilled (b) Under-filled

Figure 5: Window of Interest for Filled Level

1. converting RGB image to gray-scale
2. crop out region around the neck of the bottle as show in 5a and 5b
3. calculate the histogram of the cropped region (see fig 6, 7, and 8)
4. find the sum of the histogram of intensity values from 0 to 150, i.e the region that corresponds to the darker region of an image

5. Condition:

$$Result = \begin{cases} -1, & \text{if} \quad \sum hist_{0-150} < 3,500. \\ 1, & \text{if} \quad \sum hist_{0-150} > 5,500. \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

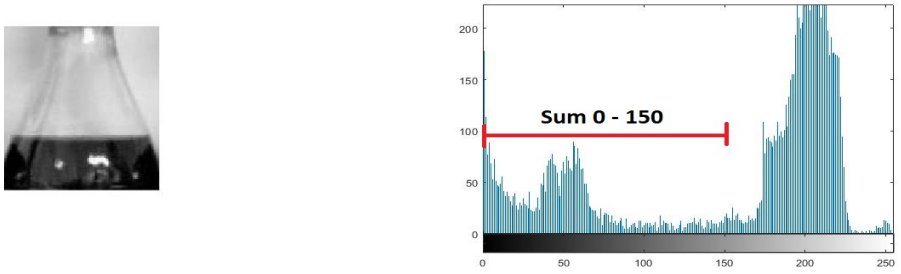
where:

(a) -1: Under-filled

(b) 1: Over-filled

(c) 0: Normal

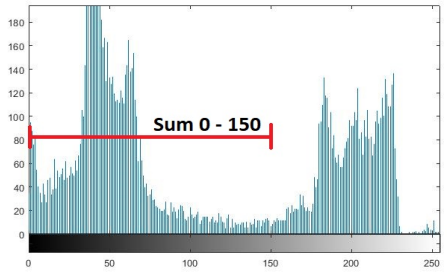
- **Reasoning**
  Once we find the histogram of the cropped region of interest, we know that the darker regions correspond to the part of the histogram closer to '0' and the brighter ones towards 255. So if we find the frequency of intensity levels in the lower region, and we find a high sum, we know it is due to the presence of darker pixels. And lower sum means means there is a higher frequency in the brighter region of the histogram.



(a) Normal Cropped (b) Histogram of Normal Filled

Figure 6: Normal: Cropped Region with Histogram

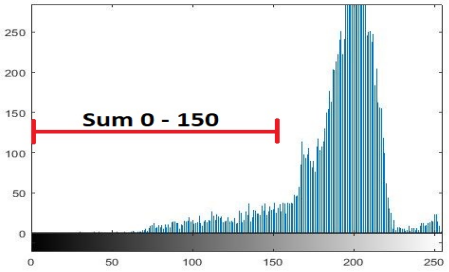
(a) Overfilled Cropped



(b) Histogram of Over Filled

Figure 7: Overfilled: Cropped Region with Histogram

(a) Under-Filled Cropped



(b) Histogram of Under Filled

Figure 8: Under-filled: Cropped Region with Histogram

Thus, when we find the sum of the histogram from the range of 0 to 150, we find the following:

- **for under-filled:** on average the sum is approximately 1,250 (due to less dark regions)
- **for over-filled:** on average the sum is approximately 9,100 (due to more dark regions)

- **Results**
  Figure 9 is a graph that summarizes the above said, taking into account *under-filled*, *over-filled*, and *normally* filled bottles:

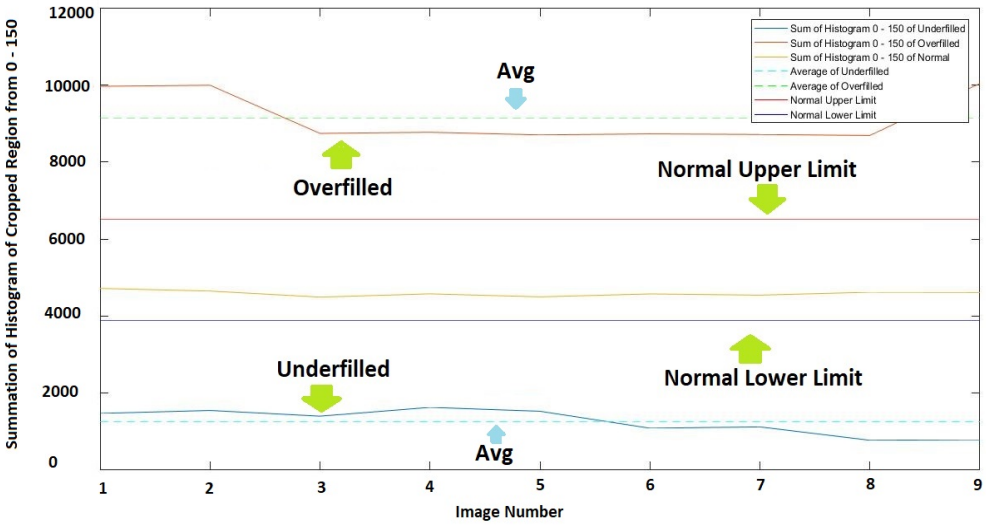


Figure 9: Comparison of Under-filled, Over-filled and Normally Filled Histogram Sum from 0 to 150

5.3 Label Present

- **Algorithm**

(a) Label Present (b) Cap Missing

Figure 10: Window of Interest

1. taking the red channel of the RGB image since it shows high intensities where the label is present. 11

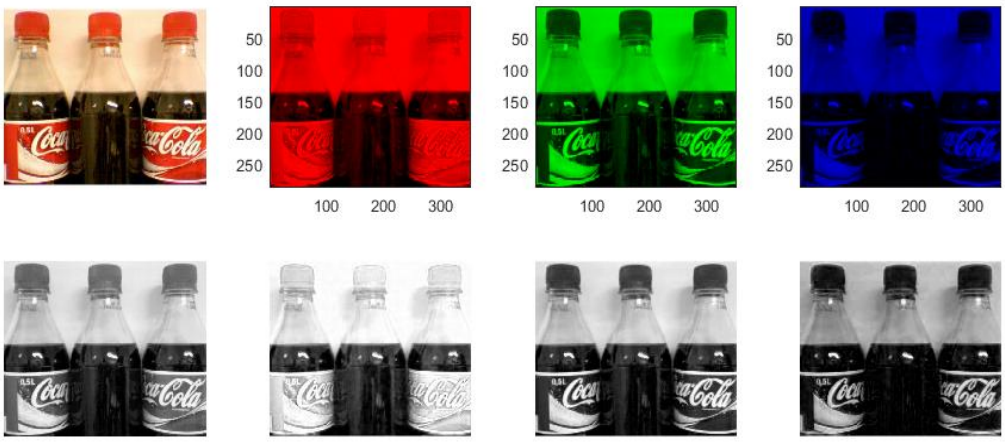


Figure 11: RGB Channels

2. cropping the region of interest which is the area where the label wraps around the bottle as shown in fig 10
3. calculate the histogram of the cropped region
4. calculate the sum of the histogram in two parts, one taking the darker intensities while the other the brighter.

$$Sum_{dark} = SumHistogram(0 - 150)$$
$$Sum_{bright} = SumHistogram(151 - 255)$$

5. Condition:

$$Result = \begin{cases} 1, & \text{if} \quad Sum_{dark} > Sum_{bright}. \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where:

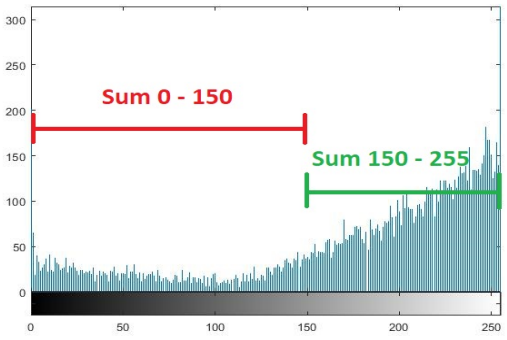
(a) 1: Label Missing

(b) 0: Label Present

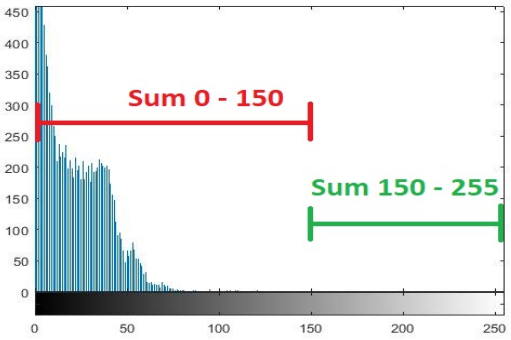
- **Reasoning**
  We consider that if we find the Sum_{dark} to be greater than Sum_{bright}, it is due to the absence of the label since the presence of a label brings in a lot of high intensity values to the image.

(a) Cropped Label Present

(b) Cropped Label Missing



(c) Histogram of Label Present



(d) Histogram of Label Missing

Figure 12: Histogram

When we see the histograms in figure 12 of the cropped section in both cases, we see the following:

- **for label missing:** the sum Sum_{dark} is way higher (due to more dark regions) with barely any counts for Sum_{bright}.
- **for label present:** the sum Sum_{dark} is less (due to less dark regions) and Sum_{bright} considerably larger.

5.4 Label Printing

- **Algorithm**

(a) Label Print Present (b) Label Print Missing

Figure 13: Window of Interest

1. taking the green channel of the RGB image since more detail is provided and contrast

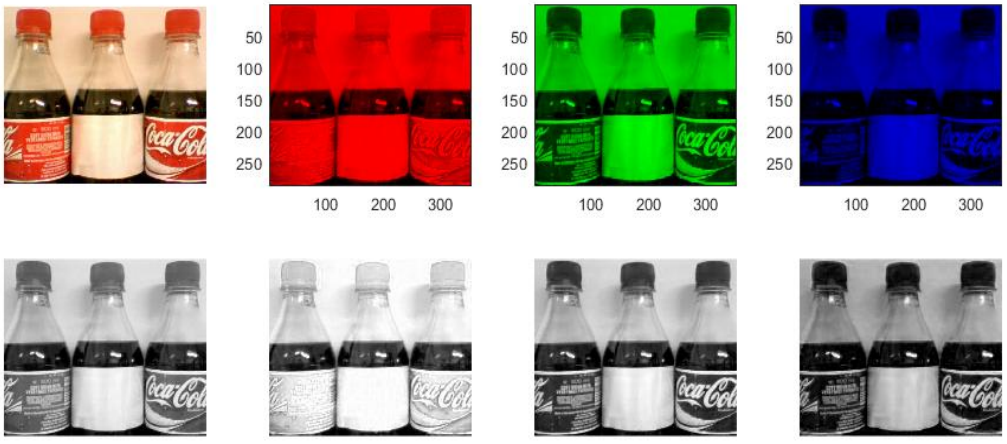


Figure 14: RBG Channels

2. cropping the region of interest as shown in fig 13, which in this case is where the label is present
3. calculate the histogram of the cropped region
4. calculate the sum of histogram in two parts, one from range *0 - 200* and *0 - 255*

$$Sum_A = SumHistogram(0 - 200)$$
$$Sum_B = SumHistogram(0 - 255)$$

5. Condition:

$$Result = \begin{cases} 1, & \text{if} \quad Sum_A/Sum_B < 0.30. \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where:

(a) 1: Label Printing Failed

(b) 0: Label Printing Successful

- **Reasoning**
  Figure 15 is the cropped images for both cases:

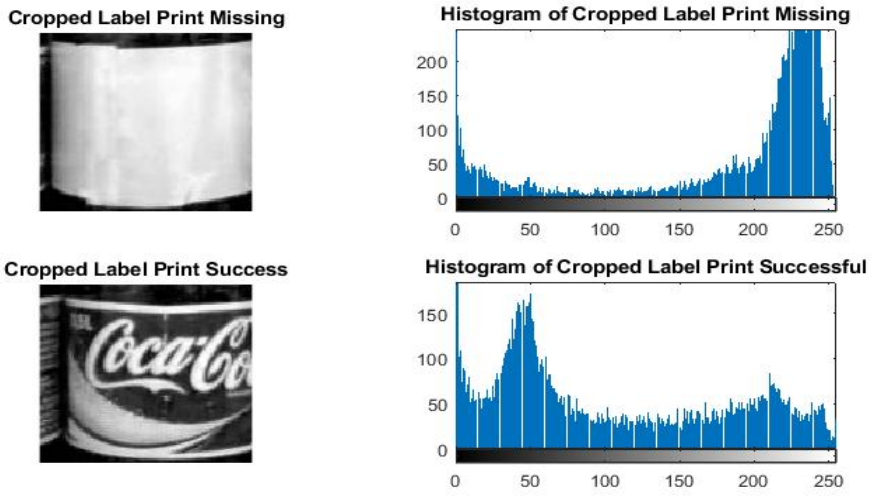


Figure 15: Histograms of Label Print Fault and Label Print Success

When we see the histogram of the cropped regions of both *Label Printing Failed* and *Label Printing Success*, we see that:

- **for label print successful:** the histogram has a higher frequency of lower intensity pixel values and lower frequency of higher intensity pixel values
- **for label print failed:** the histogram has a lower frequency of lower intensity pixel values and higher frequency of higher intensity pixel values

Thus to help differentiate between the two, we take into consideration the ratios of *Hist-Sum(0-200)* and *Hist-Sum(0-255)* for each cropped image. For a darker image (printing successful), we find the *Hist-Sum(0-200)* to be more than the *Hist-Sum(0-200)* of the brighter image (printing failed).

- **Results**

Following figure 16 is the graph depicting the ratios of *Hist-Sum(0-200)* and *Hist-Sum(0-200)* for both cases:

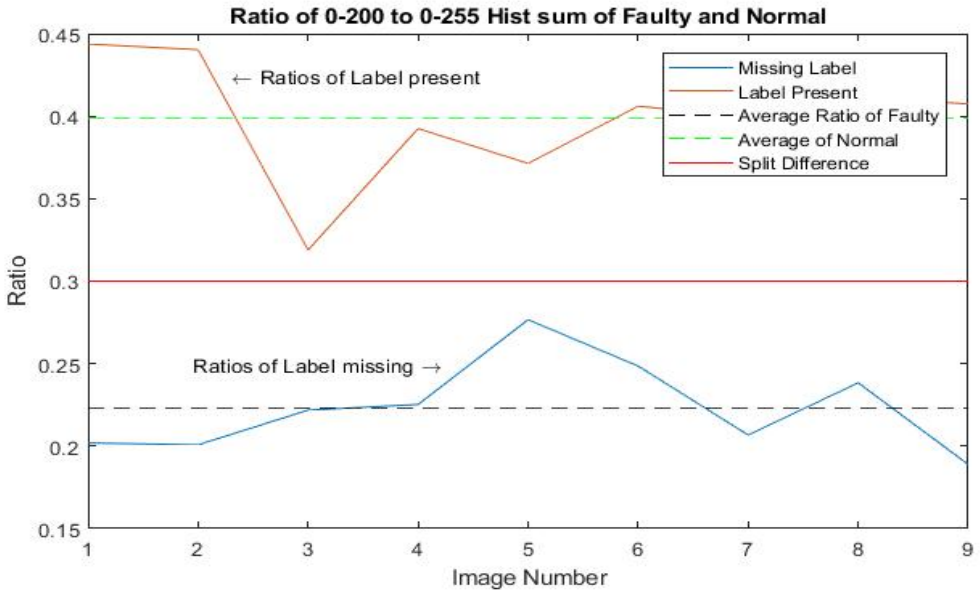


Figure 16: Ratios of Hist-Sum 0-200 to Hist-Sum 0-255 for Print Fail and Success

We see that the *red* line at *0.3* is a good separating point for both cases. Therefore, when given a ratio of *Hist-Sum(0-200)* to *Hist-Sum(0-200)* for a given cropped image, we can easily differentiate whether label printing was successful or not.

5.5 Label Not Straight

- **Algorithm**

(a) Label Straight (b) Label Not Straight

Figure 17: Window of Interest

1. select the green channel since more contrast present
2. crop the region around the label as shown in fig 17
3. apply threshold of 200 to the cropped image

$$I_{i,j} = \begin{cases} 255, & \text{if} \quad I_{i,j} > 200. \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

4. calculate the percentage of white pixels

$$Percentage_{white} = \frac{\text{Count of 255 Intensity Pixels * 100}}{\text{Total number of Pixels}}$$

5. Condition:

$$Result = \begin{cases} 1, & \text{if} \quad Percentage_{white} < 12. \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where:

(a) 1: Label Not Straight

(b) 0: Label Straight

- **Reasoning**
  Figure shows the cropped and thresholded images for both cases.

  Finding the percentage of the cropped upper white region of the label, we know the following:

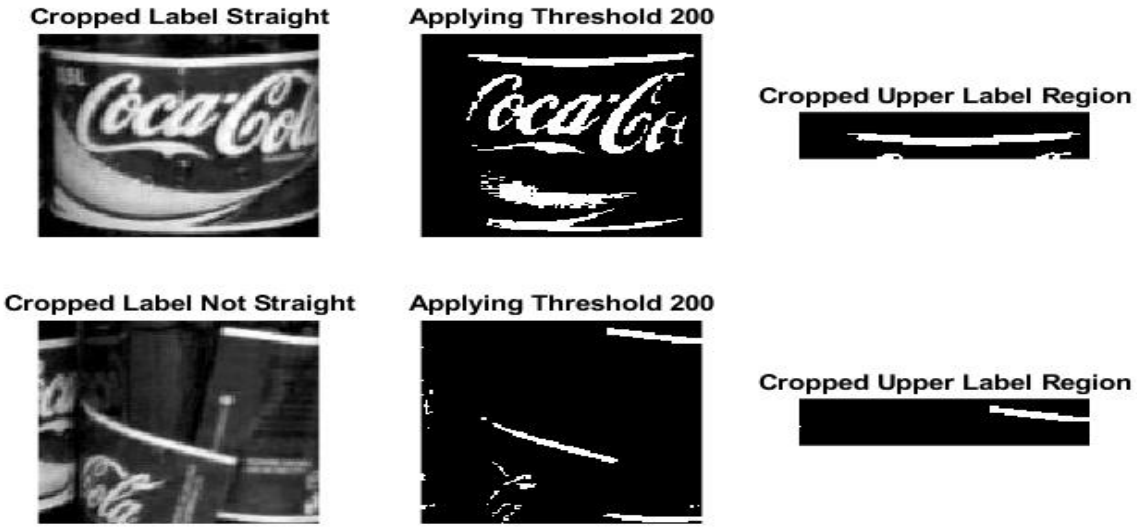


Figure 18: Cropped and Thresholded Window

- **for label straight:** there would be a higher percentage of white since the label would remain in window
- **for label not straight:** there would be a lower percentage of white since the label would go out of window

- **Results**

  Figure 19 is a graph that shows the percentages of white in cropped images of *label not straight* and *label straight* cases.

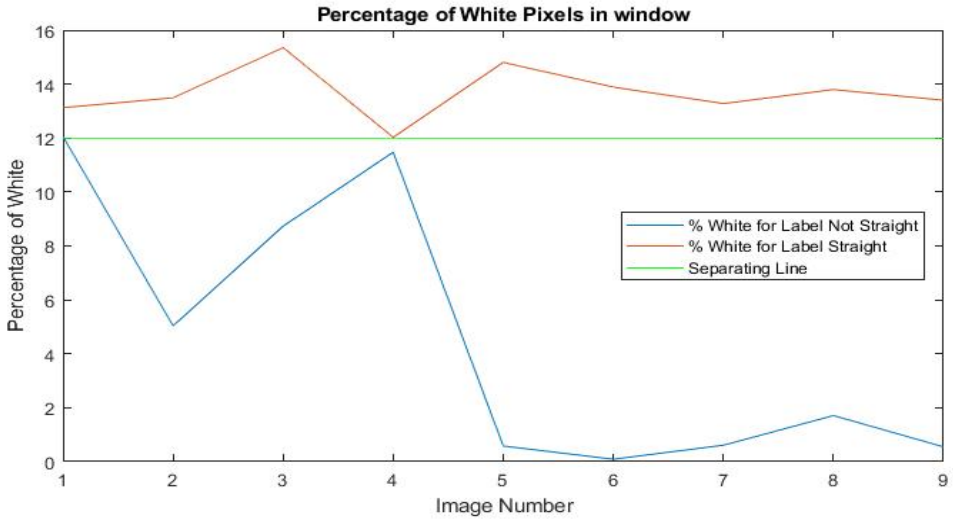


Figure 19: Percentage of White

5.6 Bottle Deformed

- **Algorithm**

(a) Normal Window (b) Deformed Window

Figure 20: Window of Interest

1. take the green channel of the RGB image
2. crop the region of interest as shown in figure 20
3. apply threshold to differentiate the dark "coke" from light "background"
4. fill any holes if present

5. remove smaller objects if present
6. use three features:
   - ratio of *Major Axis Length* to *Minor Axis Length* of an ellipse that can contain the object
   - the area of the object
   - the angles of lines on the edge of object
7. Condition
   (a) **Case 1: Normally Filled with Deformity**

$$Ratio = \frac{\text{Length of Major Axis}}{\text{Length of Minor Axis}}$$

$$Result = \begin{cases} 1, & 3 < Ratio < 4 \quad \text{and} \quad Area < 4000 \\ 0, & 3 < Ratio < 4 \quad \text{and} \quad Area > 4000 \end{cases} \tag{7}$$

   where:

   i. 1: Deformed

   ii. 0: Not Deformed

   (b) **Case 2: Over Filled with Deformity** In cases where the ratios and area are not sufficient, we find the angles of the edges. If $Angle_{line}$ be the angle of a line segment, then

$$Result = \begin{cases} 1, & \text{if exists } \ Angle_{line} < 22 \ \text{ or } \ Angle_{line} > 28. \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

   where:

   i. 1: Deformed

   ii. 0: Not Deformed

- **Reasoning**
  Once we have the filled object in the shape of the coke within the bottle, we try to fit an ellipse that will contain it. There are two possible cases, one where the deformed bottle is normally filled, and one where the bottle is overfilled. In case of under-filled, this fault detection fails since no "coke" is present to detect the shape of the bottle.

– **Case 1: Normally filled with Deformity**
See figure below which shows the ellipse fitting on a faulty and non-faulty normally filled bottle.

(a) Not Deformed (b) Deformed

Figure 21: Ellipse Fit on Normally Filled

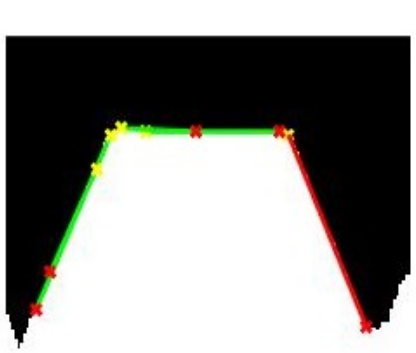
We see that when this case arise, the ratio of *Major* and *Minor Axis* does not differentiate well. So we use the *Area* of the object as the next feature. In case of a non-deformed bottle, the area is larger when compared to a deformed one as seen in figure 21. This is when the conditions specified in equation 7 are applied to make a decision.

– **Case 2: Overfilled with Deformity**

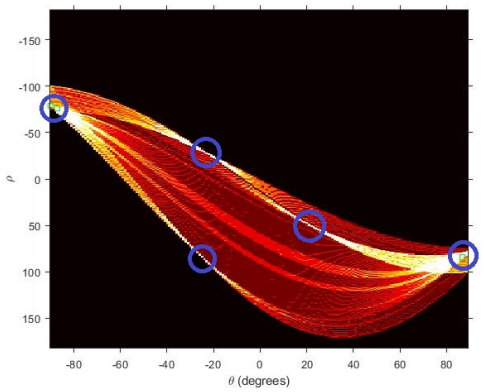
This is the case when the bottle under inspection is overfilled and needs to be checked for a deformity. We cannot use the fitting of an ellipse as a measure since the ellipse would cover a much larger area.

To find if the bottle is deformed in this case, our idea is to find the lines in image and check their angles to see if they are out of range for a non-deformed bottle. We use **Hough Transform** to achieve this.

Following is are figures to show the application of Hough Transform on the images.

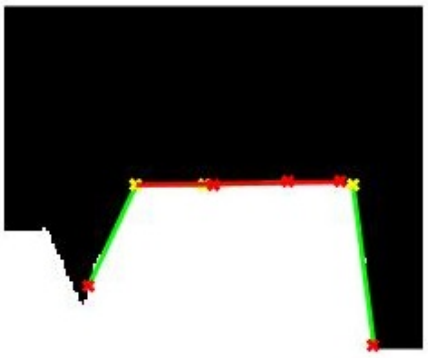


(a) Line Detection

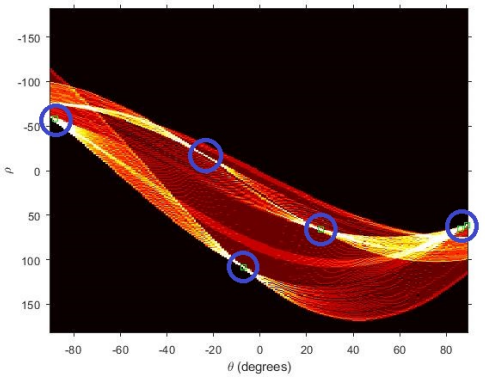


(b) Hough Lines

Figure 22: Not Deformed



(a) Lines Detection



(b) Hough Lines

Figure 23: Deformed

As can be seen in image 22a, we notice that the angles of the lines for a non-deformed image has a small range of values, which is in the range of 22 to 28 degrees from the vertical axis. From 23a, we notice that the angle is much smaller from the vertical axis (more steep).

Therefore, in order to check if the bottle is deformed in the case of overfilled, we do the following steps:

1. store all the angles of the lines generated
2. remove all angles that are 90 or close to 90 degrees from the vertical axis. So if a line is detected of +/- 5 degrees from 90, we discard them too.

3. we check if there exists a line not in the range of 22 - 28 degrees, since on average, the non-deformed bottles have edge lines approximately equal to 25 degrees from vertical axis.

There is a drawback when it comes to this approach, which is if the bottle has no coke in it, then it fails. This is because the approach is dependant on identifying the shape taken up by the coke within the bottle, which mean, if there is no coke, then there is no shape detected.

6 Drawbacks and Limitations

- The program we have created to detect faults in the images is very dependent on the intensity distribution in the image. Therefore, changes in the lighting, brightness, contrast will surely affect our system. If the lighting conditions are kept constant, then our system is very accurate, but when lighting conditions change, it may not be reliable.
- Another aspect of our approach is defining fixed windows for detecting each fault. We chose to keep a fixed window since the bottles in every image are more or less in a fixed position with very little shift in either direction. Therefore, we felt additional computation to automatically find windows on the images that are constant is not needed. We ensured that the window size is big enough to take into consideration the small shifts in the bottle positions.
- The detection of a deformed bottle has a drawback which is that is will fail for bottles that do not have coke in them. Our strategy is heavily dependant on the detection of the coke in the bottle neck. Thus, when we have an under-filled bottle with no coke visible, this particular fault detection will not work.

7 Conclusion

Our program can successfully distinguish all 100 percent of the training images for each fault. The performance is great since it provides results almost instantaneously. The reason being the usage of simple image processing techniques such as summations, histograms, etc with no computation heavy tasks.

We were able to successfully plan, design, implement, and test our software solution to the task of finding faults.