UNIVERSITY OF BURGUNDY

IMAGE PROCESSING PROJECT

MASTERS IN COMPUTER VISION AND ROBOTICS

VISUAL INSPECTION OF SOFT DRINK BOTTLE PLANT

Submitted by

PARMAR Hardiksinh

KODIPAKA Vamshi

DUDHAGARA Akshaykumar

Submitted to

Dr. Désiré Sidibé

CONTENTS

1. Abstract
2. Introduction
3. Aim
4. Methodology
   a. Bottle Cap Missing
   b. Bottle Under-Filled
   c. Label-Missing
   d. Label-Not Printed
   e. Label-Not Straight
   f. Bottle Overfilled
   g. Bottle Deformation
   h. Bottle missing
5. Analysis
6. Graphical User Interface (GUI)
7. Code Explanation
8. Results and Conclusion
9. References

ABSTRACT

Image processing is commonly utilized in industrial applications. One in every of the foremost common uses for image processing is for machine-controlled visual inspection of a product before being packaged and shipped to the client. Mistreatment of these forms of systems in production facilities aids the dodging of things wherever a faulty or substandard product is shipped to the client.

For an example situation, a Coca-Cola plant line was wont to demonstrate however image processing may be used for an automatic visual review application. A collection of pictures were provided, taken below close to constant lighting conditions within the manufacturing plant. These set of pictures embody each smart and unhealthy bottles that the bottling company expects you to use to sight sure faults as a part of the filling, capping and labelling operations before the ultimate packaging stage.

INTRODUCTION

Background the foremost common use of image processing in an industrial setting is for the automatic visual scrutiny of products leaving a production facility. Automatic scrutiny is employed to examine everything from pharmaceutical medicine to textile production. It's calculable that the bulk of product bought on food market shelves are inspected exploitation automatic "machine vision" based most systems before dispatch. Why? - to avoid the value of shipping a faulty or sub-standard item to a food market-shelf that no-one desires to buy! During this practical exercise we tend to handle a bottling production line in a very facility bottling coca-cola for the domestic market. We've a group of images, given as part of project implementation that are taken as close to constant mill lighting conditions, of the bottles as they leave the bottling line. The bottling company need a vision system to automatically determine variety of various faults which will occur throughout filling, labelling and capping stages of production in order that these bottles may be intercepted before packaging.

Figure 1 : Industrial Bottle Plant System using Visual Inspection

Our task is to style and paradigm a image processing system to sight the set of fault conditions which will occur along with distinguishing the kind of fault that has occurred. One may develop this paradigm system using Matlab.

Specified Tasks using Machine-controlled Visual Inspection :

We're needed to develop a Visible Inspection System that properly identifies each of the subsequent fault conditions which will occur within the bottling plant:

1. Bottle is missing

2. Bottle cap is missing
3. Under-filled or not stuffed in the least
4. Bottle's label missing
5. Bottle has label however label printing has failed (i.e. label is white)
6. Bottle label isn't straight
7. Bottle overfilled
8. Bottle is distorted (i.e. squashed) in a way

In every image we have a tendency to solely fascinated by classifying the central bottle within the image. One image is taken for every bottle leaving the assembly line therefore faults occurring in bottles at the edges are going to be detected individually once these specific bottles are themselves photographed central to the image. In addition, some pictures could don't have any bottle within the centre of the image – this cannot be a fault, simply a gap within the production flow stemming from a machine operating more up the line. Faults with aspect bottles and missing bottles should be unnoticed by your system – solely the eight faults on top.

Aim:

The main motto of this project is to detect the set of faults that may occur together with identifying the type of fault that has occurred in Production Line of the Bottling Plant by Visual Inspection technique using MATLAB(Image Processing Schemes).

Assumptions

Consider the below conditions as part of extraction of the images before inspection.

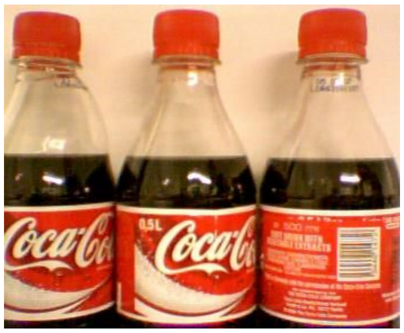
- No variation in lighting environment
- No variation in bottle positions
- Only defects in centre bottle are needed to be considered
- Defects on side bottles will not be considered in our project

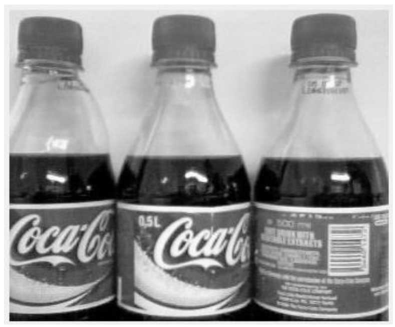
METHODOLOGY

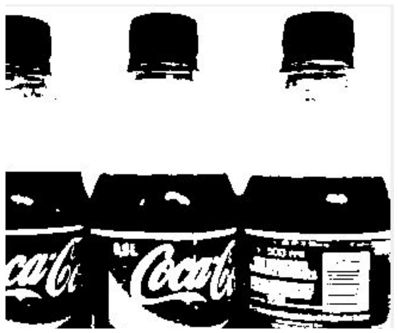
In each image we are interested in classifying the "central bottle" in the image. One image is taken for each bottle leaving the production line so faults occurring in bottles at the sides will be detected separately when these particular bottles are themselves photographed central to the image. Additionally, some images may have no bottle in the centre of the image – this is not a fault, just a gap in the production flow stemming from a machine operating further up the line. Faults with side bottles and missing bottles must be ignored by your system – only the eight faults above must be reported.

Sample Image for testing: (‘normal-image007.jpg’)

Below we read and display a no fault or normal image ‘normal-image007.jpg’ from the image set. We convert RGB image to gray scale image using matlab in-built function ‘rgb2gray’ and gray scale image to binary image using ‘imbinarize’ matlab in-built function.


RGB Image


Grey Scale Image


Binary Image

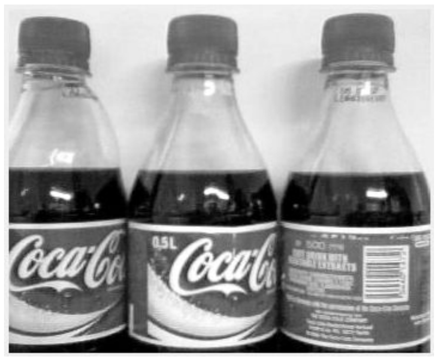

Applying imadjust on gray scale image

Figure 2 : All the above result belongs to the test image ‘normal-image007.jpg’

rgb2gray:

rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. If the input is an RGB image, it can be uint8, uint16, double, or single. The output image has the same class as the input image. If the input is a colormap, the input and output colormaps are both of class double.

imbinarize:

imbinarize Binarize grayscale 2D image or 3D volume by thresholding. BW = imbinarize(I) binarized image I with a global threshold computed using Otsu's method, which chooses the threshold to minimize the intraclass variance of the thresholded black and white pixels. BW is the output binary image.

imadjust:

imadjust adjusts image intensity values or colormap. J = imadjust(I) maps the values in intensity image I to new values in J such that 1% of data is saturated at low and high intensities of I. This increases the contrast of the output image J.

<u>Region of Interest(ROI):</u>

A Region of Interest is user defined area of an image where we can apply some filters or perform other operation too. Using the high level ROI functions, we can create many shapes and also use ROI creation classes which support properties, methods, and events that we can use to customize the behavior of the ROI.

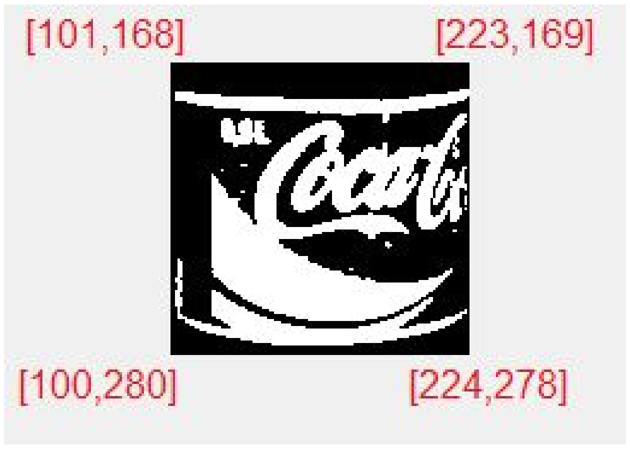


Figure 3: Applying ROI with(x1:100; y1:168, x2:123, y2: 278)

Mostly importantly, ROI is used to create a binary mask image where pixels that belong to the **ROI are set to 1 and outside pixels are set to 0**. We can define more than one ROI in an image. we can also define ROIs by intensity values where the regions are not necessarily contiguous.

We have a matlab in-built command 'imcrop' to extract the ROI. But we created our own matlab function 'Extracting_ROI' to extract ROI. It extract Region of interest associated with the image displayed in the current figure, called the target image. This function enables us to select the resizable rectangle. Then using this, we crop the target image. The cropped image, is returned. Basically this function read the input image dimensions and check if any of the coordinates of ROI are '0'. Also checks for the ROI dimensions are greater than image dimensions, then we displays:: **ooppsss...Images dimensions exceeded!**

The output of this function is :: ROI coordinates(x1,y1,x2,y2)

We select the coordinates in such a way that required region is extracted. (Keeping in mind, that x-cordinates are same in Y-direction and y-coordinates are samein X-direction).

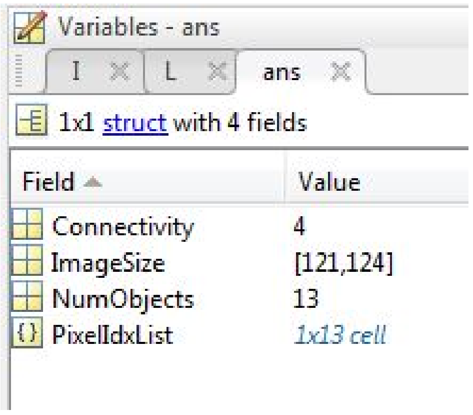
We can also perform/extract ROI using imcrop of the inbuilt function.

Bwconncomp:

Find connected components in binary image. CC = bwconncomp(BW) returns the connected components CC found in BW. BW is a binary image that can have any dimension. CC is a structure with four fields: Connectivity, ImageSize, NumObjects and PixelIdxList.

CC = bwconncomp(BW,CONN) specifies the desired connectivity for the connected components. CONN may have the following scalar values:

4	two-dimensional four-connected neighborhood
8	two-dimensional eight-connected neighborhood
6	three-dimensional six-connected neighborhood…. Etc.


Variables - ans

1x1 struct with 4 fields

Field	Value
Connectivity	4
ImageSize	[121,124]
NumObjects	13
PixelIdxList	1x13 cell

Figure 4: Applying 'bwconncomp' we set the values for connecting components

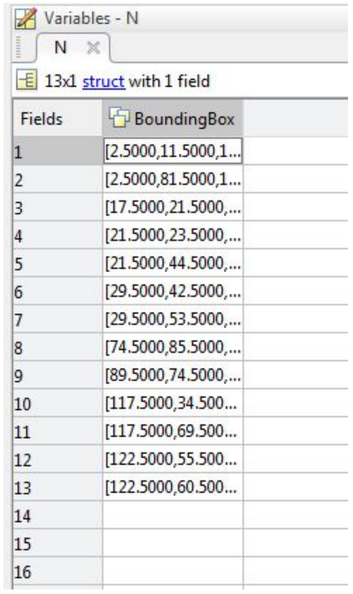
regionprops:

'regionprops' measure properties of image regions. It is a in-built command in matlab we used. STATS = regionprops(CC,PROPERTIES) measures a set of properties for each connected component (object) in CC, which is a structure returned by BWCONNCOMP. OUTPUT must be one of the following strings:: 'struct' or 'table'.

PROPERTIES can be a comma-separated list of strings or character vectors, a cell array containing strings or character vectors, 'all', or 'basic'. The set of valid measurement strings or character vectors includes: Shape Measurements like 'Area', 'BoundingBox', 'Centroid', 'ConvexHull' and Pixel Value Measurements like ''MaxIntensity', 'MeanIntensity', 'MinIntensity' etc.

Property strings or character vectors are case insensitive and can be abbreviated. If PROPERTIES is set to 'all', regionprops returns all of the Shape measurements. If called with a grayscale image, regionprops also returns Pixel value measurements. If PROPERTIES is not specified or if it is set to 'basic', these measurements are computed: 'Area', 'Centroid', and 'BoundingBox'.

regionprops can be used on contiguous regions and discontiguous regions. Contiguous regions are also called "objects", "connected components" and "blobs". If the first input is BW, BW must be a logical array and it can have any dimension. If the first input is CC, CC must be a structure returned by BWCONNCOMP.



13x1 struct with 1 field

Fields	BoundingBox
1	[2.5000,11.5000,1...
2	[2.5000,81.5000,1...
3	[17.5000,21.5000,...
4	[21.5000,23.5000,...
5	[21.5000,44.5000,...
6	[29.5000,42.5000,...
7	[29.5000,53.5000,...
8	[74.5000,85.5000,...
9	[89.5000,74.5000,...
10	[117.5000,34.500...
11	[117.5000,69.500...
12	[122.5000,55.500...
13	[122.5000,60.500...
14	
15	
16	

Figure 5: Applying regionprops with property as 'BoundingBox' and we get struct output as above.

NOTE: *We explained and performed all the concepts above on the 'normal' or 'no fault image' i.e; 'normal-image007.jpg' for testing. All these concepts above are used in one or all of the functions as per our requirement for fault analysis.*

Blob Pixel Unification and counting targets

A blob means a small lump for analyzing images that have been converted with binary processing, for easier differentiation we will use Blob analysis. In presence inspection, this means an aggregation of pixels having the same shade value. Suppose there are two white sections (targets or blobs) after binary processing. Counting the number of these white sections is one of the basic methods of presence inspection. Check for surrounding neighbour pixels in-order to unify or exclude the target white region.

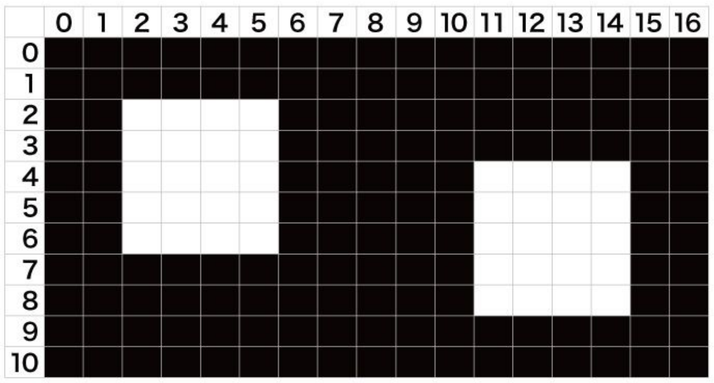


Figure 6: Blob Analysis of Neighbor pixels around ROI

In the following example, the original image is converted with binary processing and white lumps are counted based on the area size. This is another blob analysis.

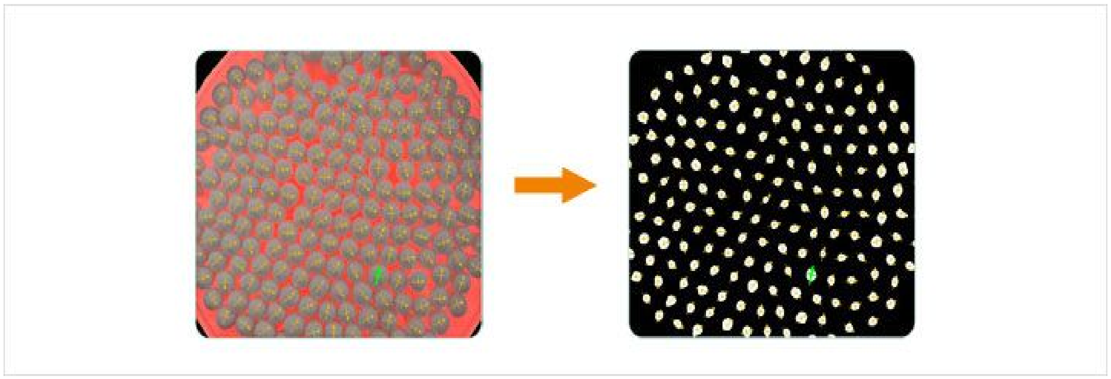


Figure 7: Counting Similar Pixels for % Black Intensities

Threshold Setting:

We convert grayscale into images that consist of two shade levels, black and white by binary processing. Set predefined specific threshold which converts to black or white pixels. (Converts into white if a value > threshold and converts into black if a value falls < threshold.)

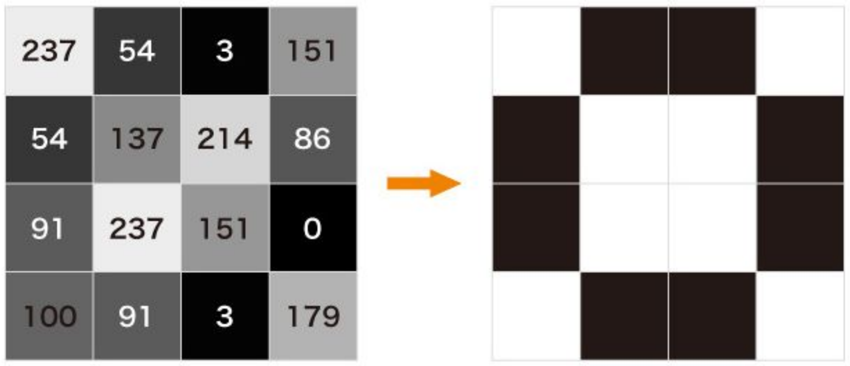


Figure 8 : Pixel Connectivity and Threshold Setting

A. Bottle Cap Missing

For cap detection method, we will define a horizontal reference and level detection line using ROI and accumulate all pixels and find the black pixel percentage, that is from Line1 and Line2 for example as shown in fig. To achieve an accurate result, we definitely keep these expected three lines separate for their purpose.

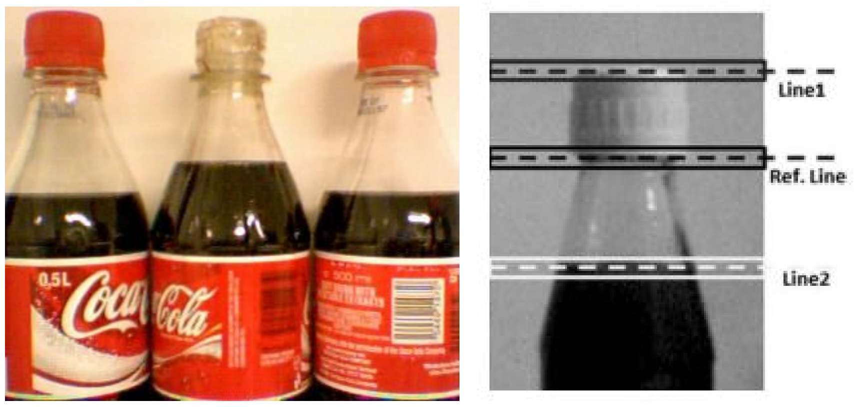


Figure 9 : Cap missing & Cap Detection Bottle Image

Use these 3 lines to fix the cap position and keeping exactly at the lowest groove of the bottle neck. Using RGB, we can detect the color of bottle cap and then its position is compared with the base reference line.

B. Bottle Under-Filled

First we will define a horizontal reference and level detection line. To achieve this, we observe the level detection line to be below the reference line using ROI and accumulate all pixels and find the black pixel percentage.

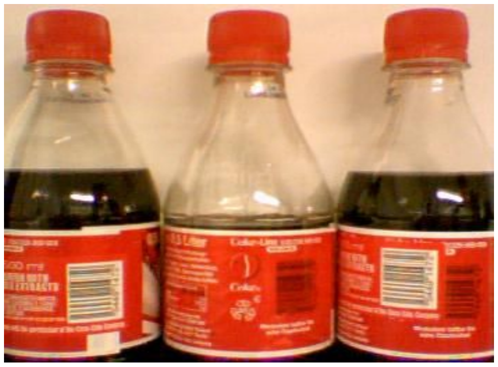


Figure 10 : Under-Filled Bottle

We will take first horizontal line below the cap as a line1 and line2 over the label. we set the reference line with respect to the liquid level of the side bottle.

Using RGB method, we can detect the liquid level in the bottle and then its level is compared with the base reference line. Scan along the reference line then extract the no of black pixels and observe it with the threshold limit set for the reference. If it is less than threshold limit(here in our case: <25) then the bottle is under filled.

C. Label-Missing

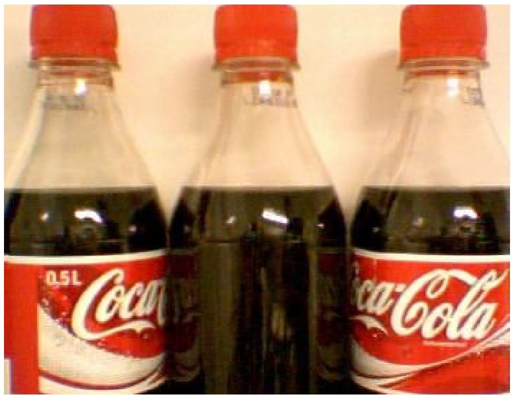


Figure 11: Label-Missing Bottle

Taking RGB scale, the intensity of label are set to binary values which results in 0’s and 1’s. Then we count number of black and white pixels and compare it with the predefined threshold which is close to the black average intensity level with respect to the region of interest over centered bottle. Observe if sum of the pixels and find the percentage(here in our case >50), then consider it to be Label Missing.

D. Label-Not Printed

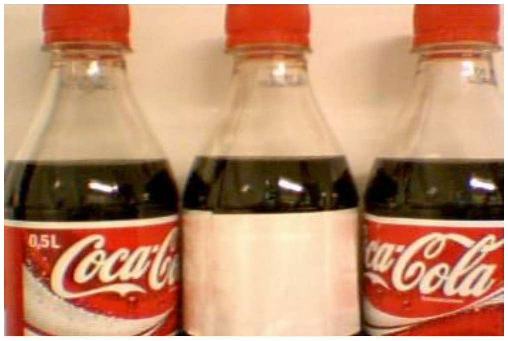


Figure 12 : Label not Printed bottle

Taking RGB scale, the intensity of label are set to binary values and does the same process as like in 'Label Missing'. Observe if sum of the pixels and find the percentage(here in our case <50), then consider it to be Label Not Printed.

E. Label-Not Straight

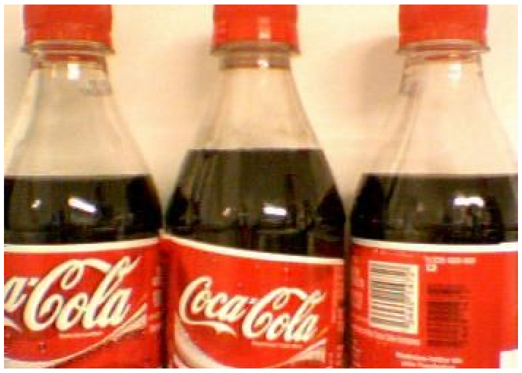


Figure 13 : Label not Straight bottle

After top and bottom inspection of empty bottle the liquid is filled and cap is placed on the bottle. To reduce the computational complexity, the selected image is converted to gray scale. The basic idea is to detect edges lines, so as edge detection algorithm such as Sobel or Canny can be applied. To detect sharp edges **'Sobel edge detection'** is used where it convolves the impedance image with the bounding box. Set the bounding box's height and width in a such a way that it matches with the region of interest's no. of pixels(within limits; here in our case:: Consdition1: maximum_Width <= 100 || maximum_Height >= 10; Condition2: Percentage_black >= 13;), then it is treated as label straight else the label is not straight.

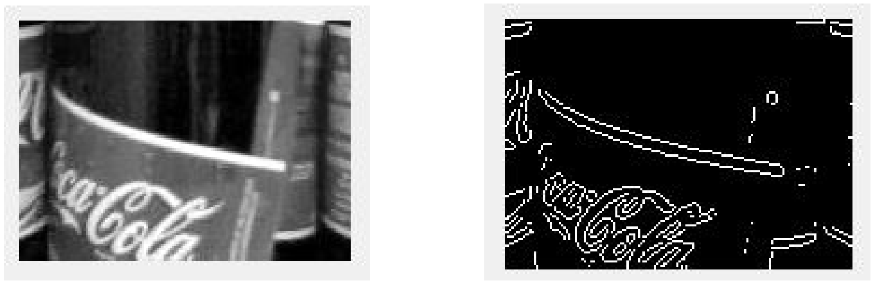


Figure 14: Result of Edge Detection(inbuilt edge() function): for Label Not Straight

Find edges in intensity image -->for this fault detection only.

Edge takes an intensity or a binary image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere. Edge supports six different edge-finding methods:

The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

Sobel Method::

BW = edge(I,'sobel') specifies the Sobel method. BW = edge(I,'sobel',THRESH) specifies the sensitivity threshold for the Sobel method. edge ignores all edges that are not stronger than THRESH. If you do not specify THRESH, or if THRESH is empty ([]), edge chooses the value automatically.

F. Bottle OverFilled

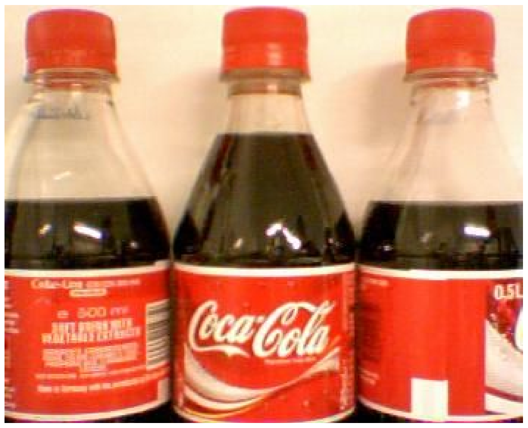


Figure 15 : Over-Filled Bottle

We do all the process same as Bottle Underfilled then, using RGB method, we can detect the liquid level in the bottle and then its level is compared with the base reference line. Scan along the reference line then extract the no of black pixels and observe it with the threshold limit set for the reference. If it is less than threshold limit(here in our case: >40) then the bottle is over-filled.

G. Bottle Deformation

Taking Region of interest ROI, set the image intensity to the saturation levels. Using region property 'Bounding Box', calculate the maximum binary pixels intensities while scanning horizontally and set them as a maximum width. Also scan vertically and observe the maximum height then compute the area. Now set the both definite limits: upper limit and lower limit.

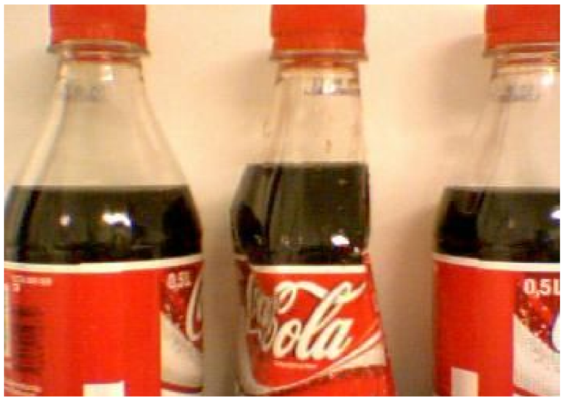


Figure 16 : Deformation Bottle

Sum of absolute difference and correlation:

While scanning, horizontally and vertically size of the bounding box varies. we compute the max area within the region R1 and update it with the newly computed area R2 extending in max height and max width then we take the difference between the region R1 and region R2 which is then called as absolute difference. We then add them up to give a area of difference in region. Correlation gradient varies with size of the bounding box/masked pixels along both directions so defined as maxarea.

We set the defined limits::

```
   maximum_Area_Result =   (maximum_Area >= 9800) && (maximum_Area <= 12000);
   maximum_Area_WResult = (maximum_Area_W >= 110) && (maximum_Area_W <= 130);
   maximum_Area_HResult = (maximum_Area_H >= 80) && (maximum_Area_H <= 100);
Result=maximum_Area_Result && maximum_Area_WResult && maximum_Area_HResult
```

(gives logical 0 or 1).

This logic of 0 or 1 gives deformation logic to be used in our main/GUI function(Soft_Drink_Bottling_Plant.m) file.

H. Bottle Missing

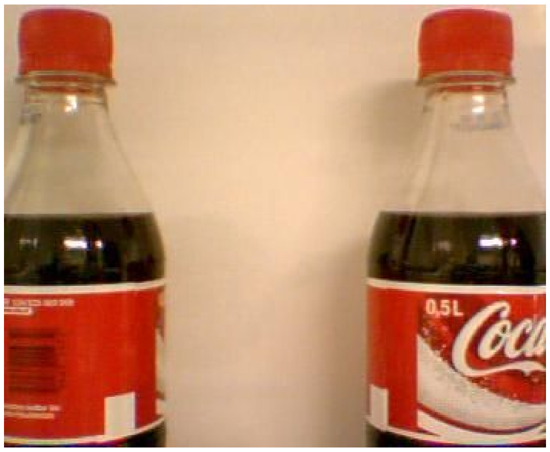


Figure 17 : Bottle missing

To start the process of checking the faults, begins with finding whether the Bottle is present or not. If Bottle is not present, display the bottle is missing else begin the procedure of fault analysis. Observe, if black pixel percentage is less than threshold limit(here in our case: <10) then the bottle is bottle missing.

ANALYSIS

The Coca-Cola logo which is very typical and have a characteristic, scale-invariant signature in the frequency domain, particularly in the red channel of RGB. the pattern of red-white-red come across by a horizontal scan line and will have a individual “flow” as it passes through the central axis of the logo. “flow” will go fast or slow at different scales but will remain proportionally equivalent. we could identify/define many scanlines, horizontally and vertically through the logo. Call these the "signature scan lines."

To using a Scan of the image in the horizontal strips, we can find this signature in the target image. If we know the high frequency in the red channel and if it is followed by one of the frequency flow identified in the training session and if it matches then we will know the scan line’s orientation and location in the logo.

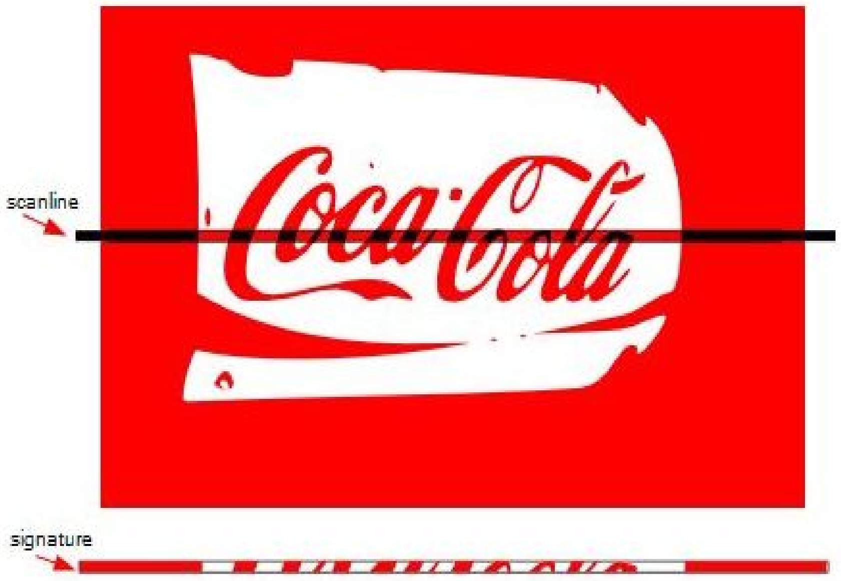


Figure 18 : Example Image for Region of Interest :: Horizontal Scanning

For Bottle Recognition, we would look for coke (the brown liquid) adjacent to the logo -- that is, inside the bottle. Or, in the case of an empty bottle, I would look for a cap which will always have the same basic shape, size, and distance from the logo and will typically be all white or red. Search for a solid color elliptical shape where a cap should be, relative to the logo. Not false proof of course, but our goal here should be to find the easy ones fast.

MATLAB Implementation

1. Database Management (Images as inputs)

   All the fault image dataset are given to the path as below and We will give the path of images as a input. Out of images dataset, select a image and analyze the fault.

Path:: "D:\E\IP\IP-project-2018\TrainingData\Visual inspection system\underfilled-image031.jpg"

Image Data Path Specification:

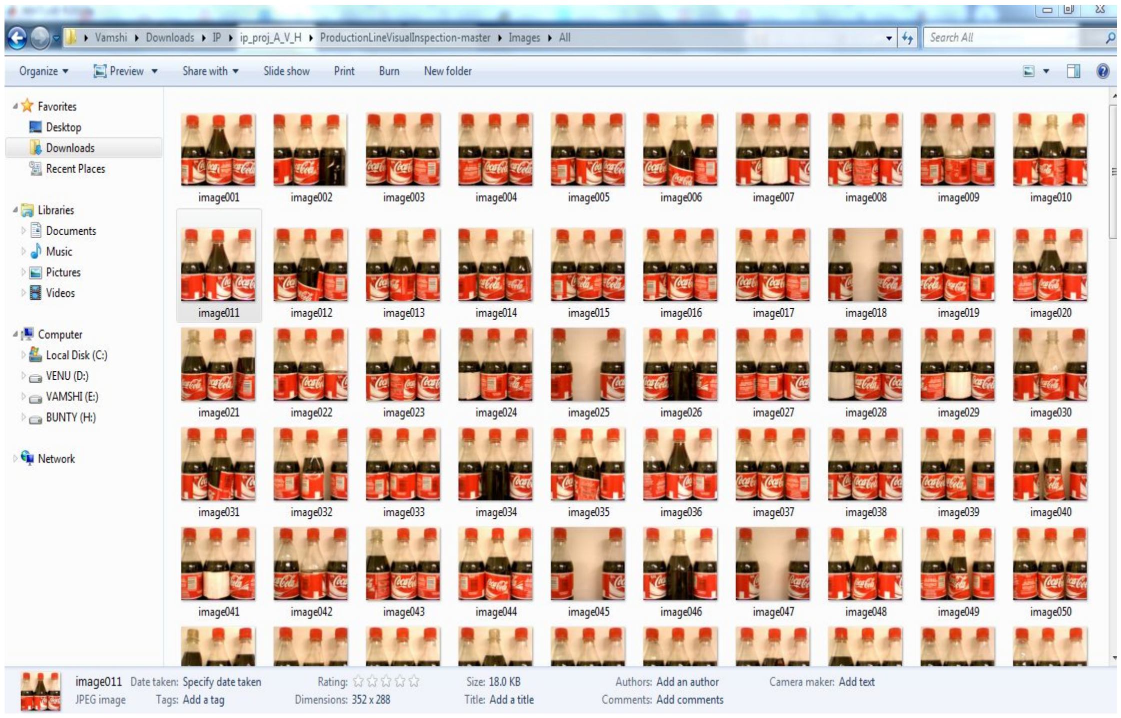


Figure 19: Data set for Image input

Graphical User Interface(GUI)

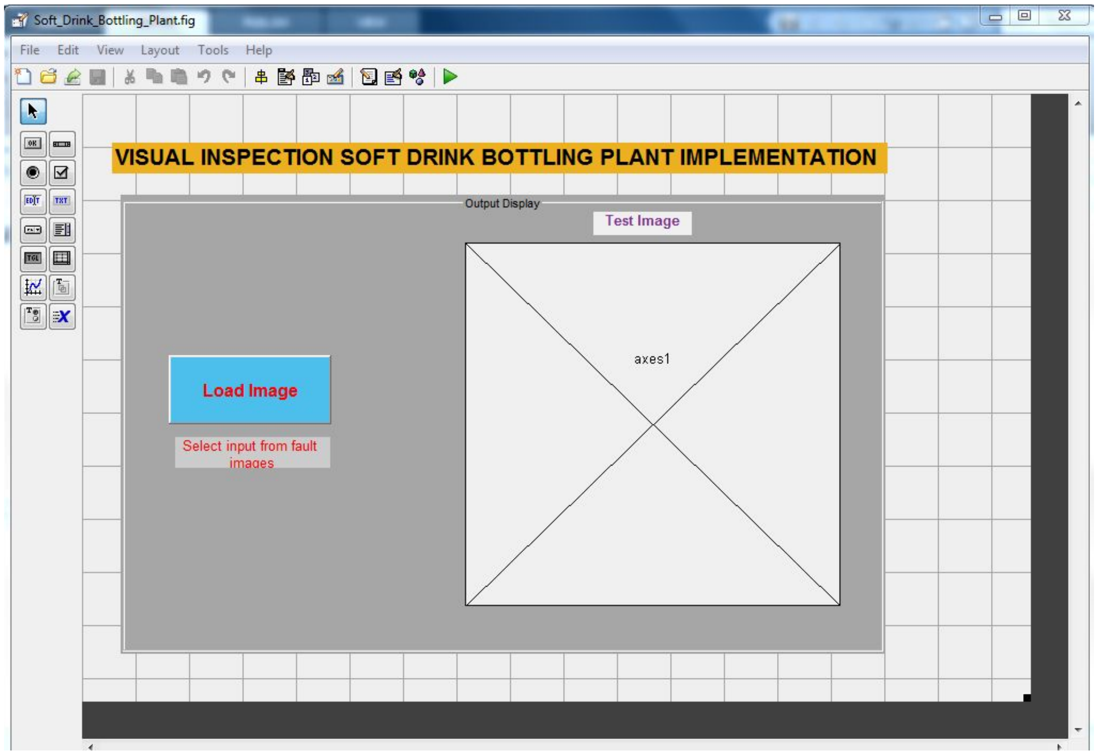

Soft_Drink_Bottling_Plant.fig
File Edit View Layout Tools Help
VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
axes1
Load Image
Select input from fault images

Figure 20 : construction of GUI

Explanation about GUI:

1. **Push button :** Use this GUI component to Load the input image from the image dataset.
2. **Axes:** To display the image with proper dimensions and size.
3. **Text box:** To display the instruction in text format.
4. **Button Group:** To arrange the components in the proper order
5. **Pop-up box:** To display the output result/type of the fault.

EXPLANATION OF CODE :: FUNCTION DESCRIPTION

A. Bottle missing

Function Implementation:

1.Read the image from the input dataset.

2.Convert the RGB image to gray scale image using MATLAB inbuilt function 'rgb2gray'.

3.Using ExtractROI.m read the region of interest.

I. Read four coordinate manually and name them as x1,y1,x2,y2 as inputs to extract ROI.

roi = ExtractROI(image, 1, 135, 250, 225);

II. Check those points if its zeros then display the error message as:

"[ERROR]: Ooops you forgot MATLAB indices start at 1!"

III. Read the Image dimensions and compare it with the inputs of Extract ROI function. If less than those of inputs x1,y1,x2,y2 then display the error as:

"[ERROR]: Images dimensions (%d, %d) exceeded!\n"

IV. Give the output of this function as region of interest(assign variable as ROI).

4. Convert the gray scale image to Binary and set the threshold as 150 among 0-255.

5. Compute the Average no of black pixels and take the percentage.

6. Observe weather it is less than 10. If yes then Bottle is missing.

7. Else:: "oppsss….Bottle is Missing" then Continue

B. Cap Missing

Function Implementation:

1. Repeat the step 1 and 2 as like in Bottle missing. Now input the coordinates x1,y1,x2,y2 as:

roi = ExtractROI(image, 5, 150, 45, 200);

Input :: y1=5, x1=150, y2=45, x2=200

2. Repeat the next steps as like in bottle missing.

3. Observe the black pixel percentage less than 25.

4. Else:: Now check cap is missing if yes: "oppsss….Bottle Cap is Missing"

C.Bottle underfilled

Function Implementation:

1. Repeat the step 1 and 2 as like in Bottle missing. Now input the coordinates x1,y1,x2,y2 as:

    roi = ExtractROI(image, 130, 140, 170, 220);

    Input :: y1=130, x1=140, y2=170, x2=220 % For Ideal liquid level

2. Repeat the next steps as like in bottle missing.
3. Observe the black pixel percentage less than 25.
4. Else:: Now check Bottle is underfilled, if yes print: “oppsss….Bottle is underfilled”

D.Label missing

Function Implementation:

1. Repeat the step 1 and 2 as like in Bottle missing. Now input the coordinates x1,y1,x2,y2 as:

    roi = ExtractROI(image, 180, 110, 280, 240);

    Input :: y1=180, x1=110, y2=280, x2=240

2. Repeat the next steps as like in bottle missing.
3. Observe the black pixel percentage greater than 50.
4. Else:: Now check label is missing, if yes print: “oppsss….Label is missing”

E.Label not printed

Function Implementation:

1. Repeat the step 1 and 2 as like in Bottle missing. Now input the coordinates x1,y1,x2,y2 as:

    roi = ExtractROI(image, 180, 110, 280, 240);

    Input :: y1=180, x1=110, y2=280, x2=240

2. Repeat the next steps as like in bottle missing.
3. Observe the black pixel percentage less than 50.
4. Else:: Now check label is not printed, if yes print: “oppsss….Label is not printed”

<u>F.Label not straight</u>

<u>Function Implementation:</u>

1.Read the image from the input dataset.

2.Convert the RGB image to gray scale image using MATLAB inbuilt function ‘rgb2gray’ 3.adjust the intensity values over the threshold to white pixels and intensity values below the threshold are set to black pixels..

4.Using ExtractROI.m read the region of interest.

I. Read four coordinate manually and name them as x1,y1,x2,y2 as inputs to extract ROI.

roi = ExtractROI(image, 170, 110, 195, 250);

Input :: y1=170, x1=110, y2=195, x2=250

5. Perform Edge detection using “Sobel edge detector” resize it to single dimension with reduce magnitude and name it as edge of the region.

6. Now Calculate the no of binary connected pixels around the ROI using ‘bwconncomp’ with region properties as bounding box.

7. Loop over all bounding boxes and compute max height and max width around the ROI.

8. Using ExtractROI.m read the region of interest.

I. Read four coordinate manually and name them as x1,y1,x2,y2 as inputs to extract ROI.

roi = ExtractROI(image, 180, 110, 230, 250);   % top half of the label

Input :: y1=180, x1=110, y2=230, x2=250

9. Convert ROI into binary image with threshold 50.

10. Calculate the percentage of black pixels.

11. Observe max width is <= 100 and max height is > 10.give the result as horizontal limit(white).

12. Check percentage of black pixels >= 30.

13.compare horizontal limit and black pixel percentage and return as final result.

14. Else:: Now check label is not straight, if yes print: “oppsss….label is not straight”

<u>G.Bottle deformed</u>

<u>Function Implementation:</u>

1.Read the image from the input dataset.

2.Convert the RGB image to gray scale image using MATLAB inbuilt function ‘rgb2gray’ 3. adjust the intensity values over the threshold to white pixels and intensity values below the threshold are set to black pixels..

4.Using ExtractROI.m read the region of interest.

I. Read four coordinate manually and name them as x1,y1,x2,y2 as inputs to extract ROI.

roi = ExtractROI(image, 190, 100, 280, 260);

Input :: y1=190, x1=100, y2=280, x2=260

5**.** Now binarize red channel and give the threshold value as 200.

6. Use the mask of binary connecting component ‘Bounding box’ from the regionprops function.
7. Compute the area using maxH, maxW and loop over all the regionprops and update the maxArea.
8. Now verify the definite limits for maxArea >= 9800 and maxArea <= 12000.
9. Observe the maxAreaW>= 110 and maxAreaW <= 130.
10. Observe the maxAreaH>= 80 and maxAreaH <= 100.
11. Obtain the result (if not true) by comparing maxArearesult, maxAreaHresult and maxAreaWResult and return result as function output.
12. Else:: Now check bottle is deformed, if yes print: “oppsss….Bottle is deformed”

<u>H.Bottle overfilled</u>

<u>Function Implementation:</u>

1. Repeat the step 1 and 2 as like in Bottle missing. Now input the coordinates x1,y1,x2,y2 as:

roi = ExtractROI(image, 110, 140, 140, 220);

Input :: y1=110, x1=140, y2=140, x2=220

2. Repeat the next steps as like in bottle missing.

3. Observe the black pixel percentage greater than 40.

4. Else:: Now check bottle is overfilled, if yes print: “oppsss….Bottle is overfilled”

If none of the above condition is satisfied, print “No fault detected”.

Execution Flowchart:

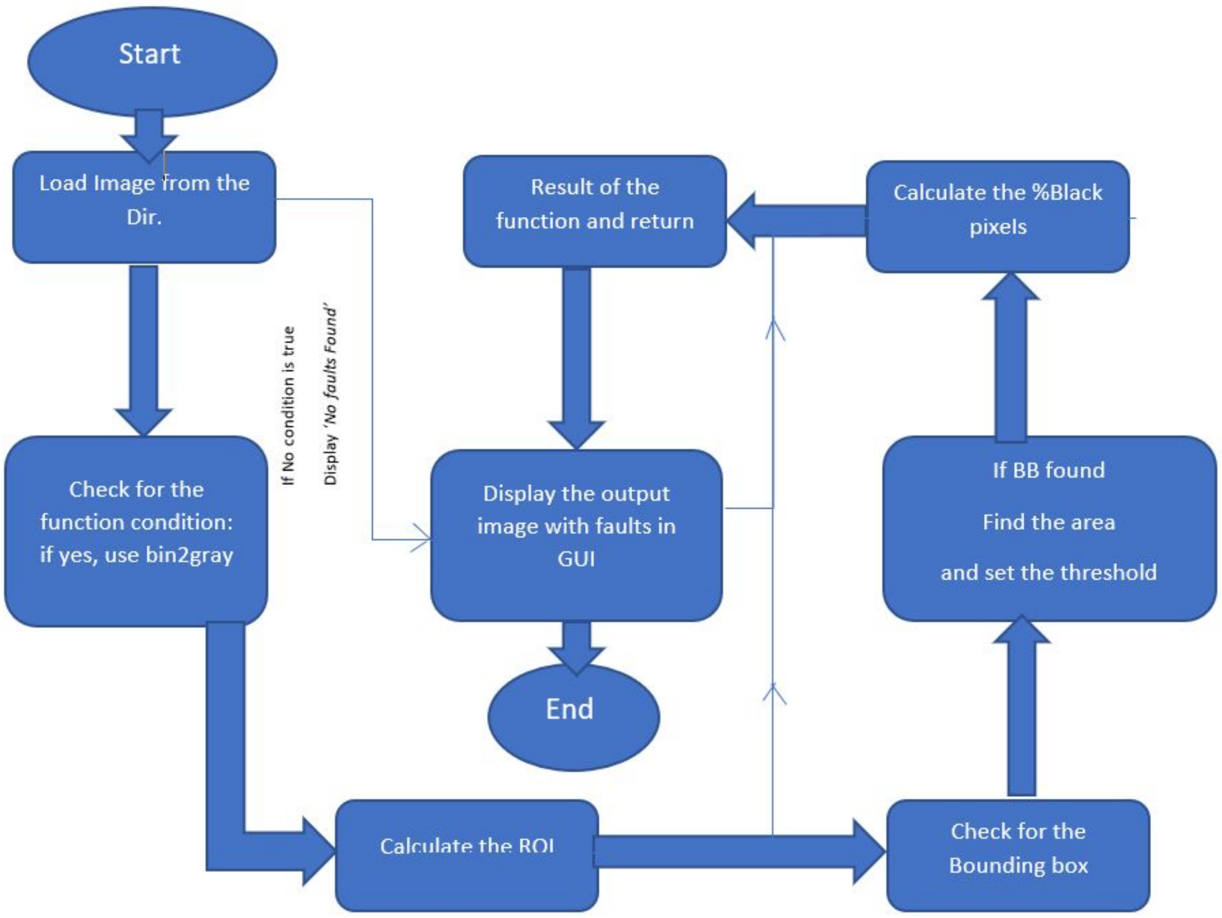
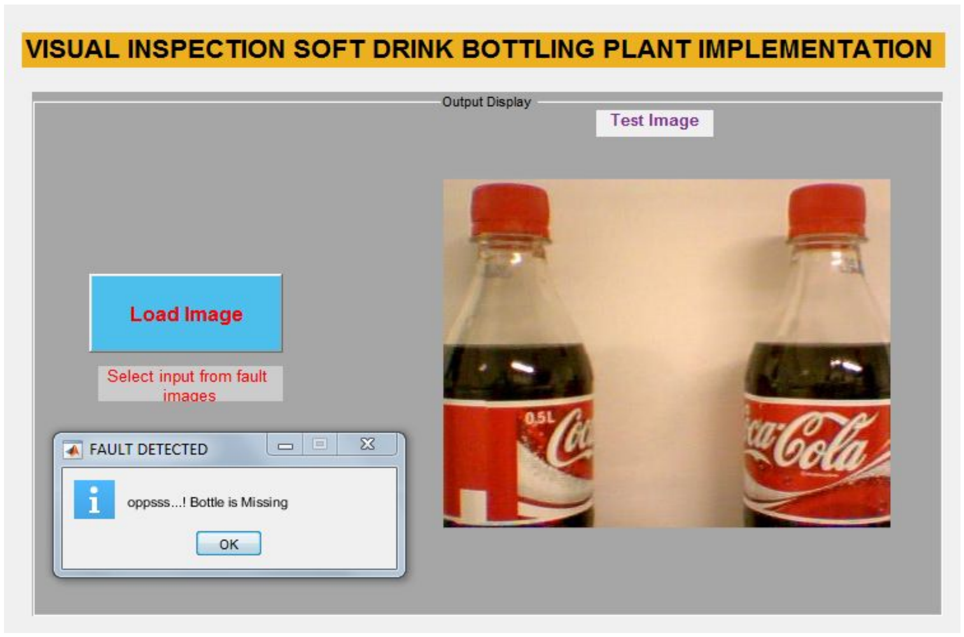

Start
Load Image from the Dir.
Check for the function condition: if yes, use bin2gray
If No condition is true
Display 'No faults Found'
Calculate the ROI
Check for the Bounding box
If BB found
Find the area
and set the threshold
Calculate the %Black pixels
Result of the function and return
Display the output image with faults in GUI
End

Figure 21: FlowChart for Execution of Fault Analysis

OUTPUTS/RESULTS

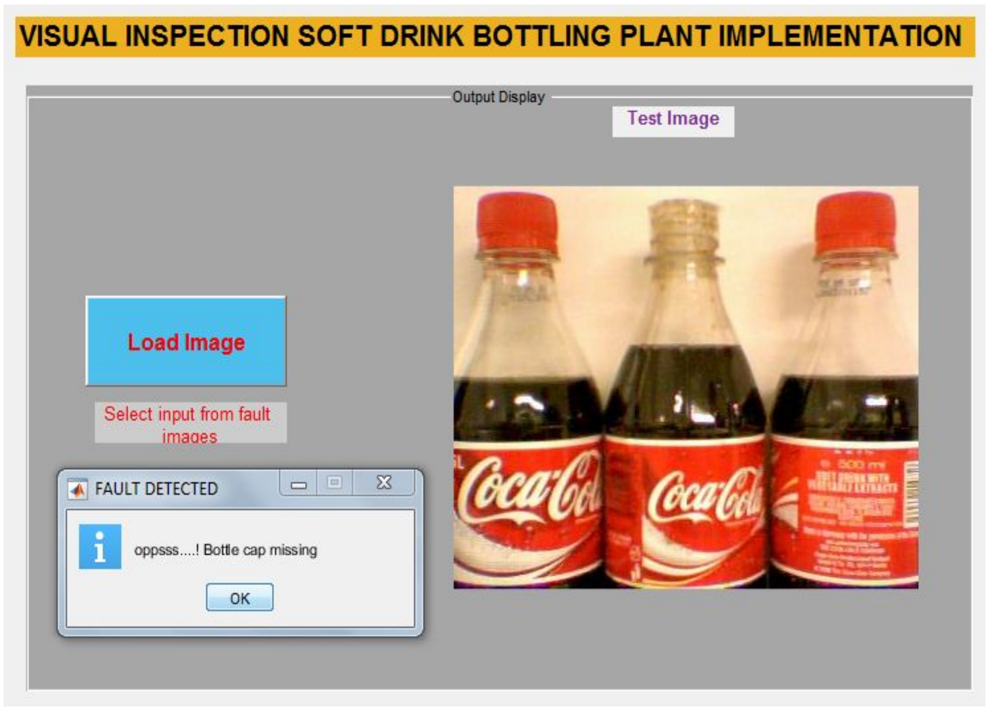
1. Bottle is Missing:


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
Load Image
Select input from fault images
FAULT DETECTED
OK

oppsss...! Bottle is Missing

GUI Display for Bottle Missing Fault

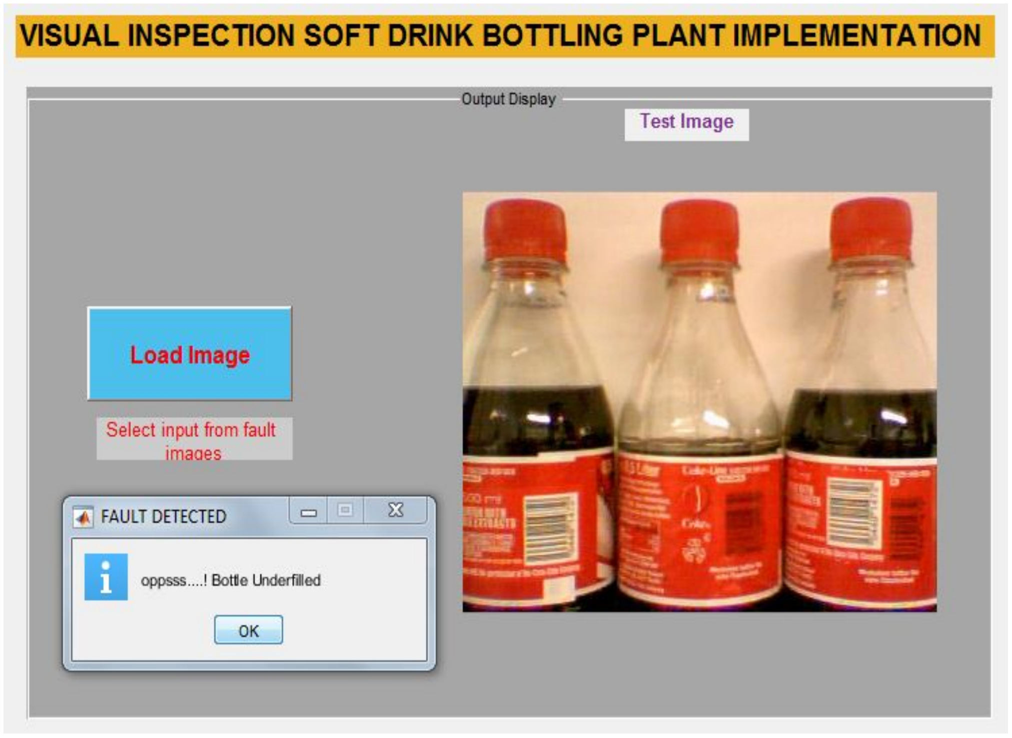
2.**Bottle cap is Missing:**


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
Load Image
Select input from fault images
FAULT DETECTED
OK

oppsss....! Bottle cap missing

GUI Display for Bottle Cap Missing Fault

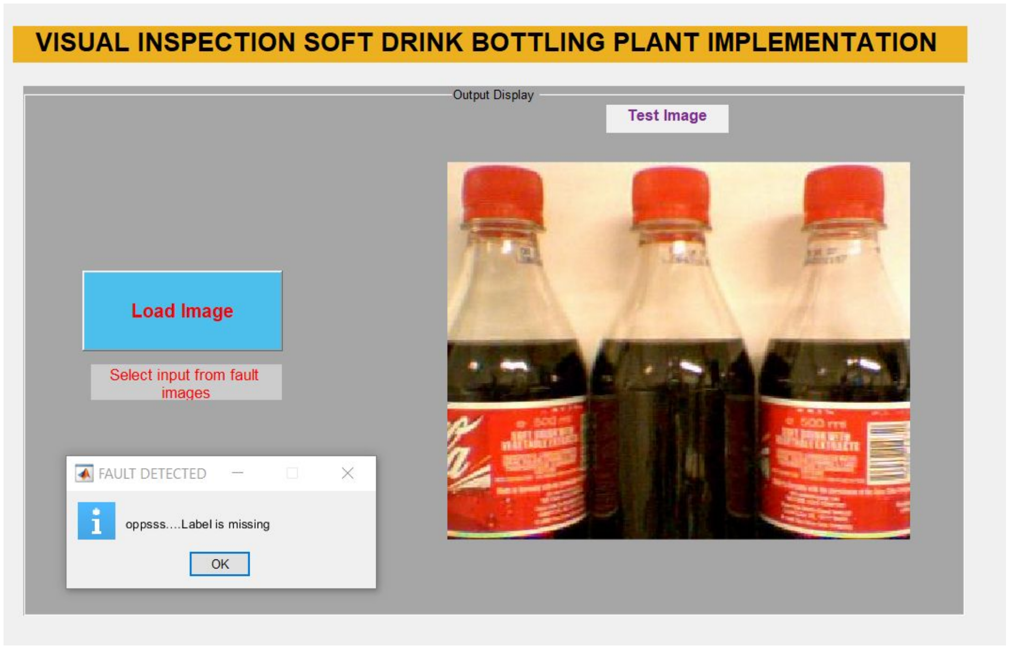
3. Bottle underfilled:


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
Load Image
Select input from fault images
FAULT DETECTED
OK

oppsss....! Bottle Underfilled

GUI Display for Under Filled Fault

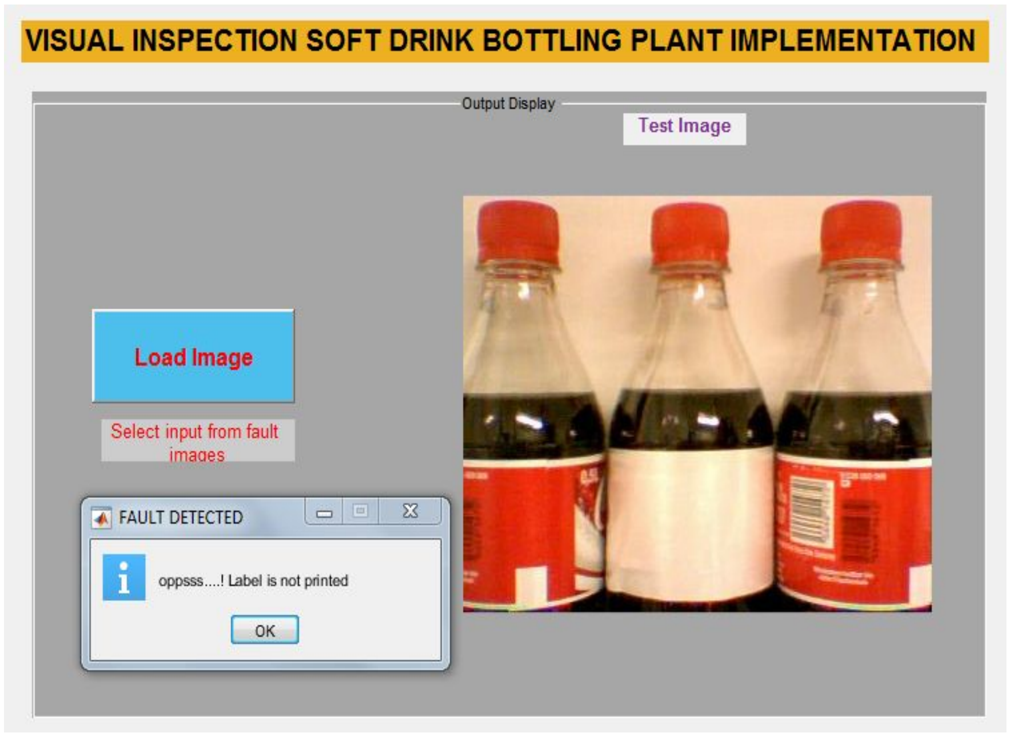
4. Label is Missing:


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
Load Image
Select input from fault images
FAULT DETECTED
OK

oppsss....Label is missing

GUI Display for Label Missing Fault

5. Label is not printed:


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION

Output Display

Test Image

Load Image

Select input from fault images

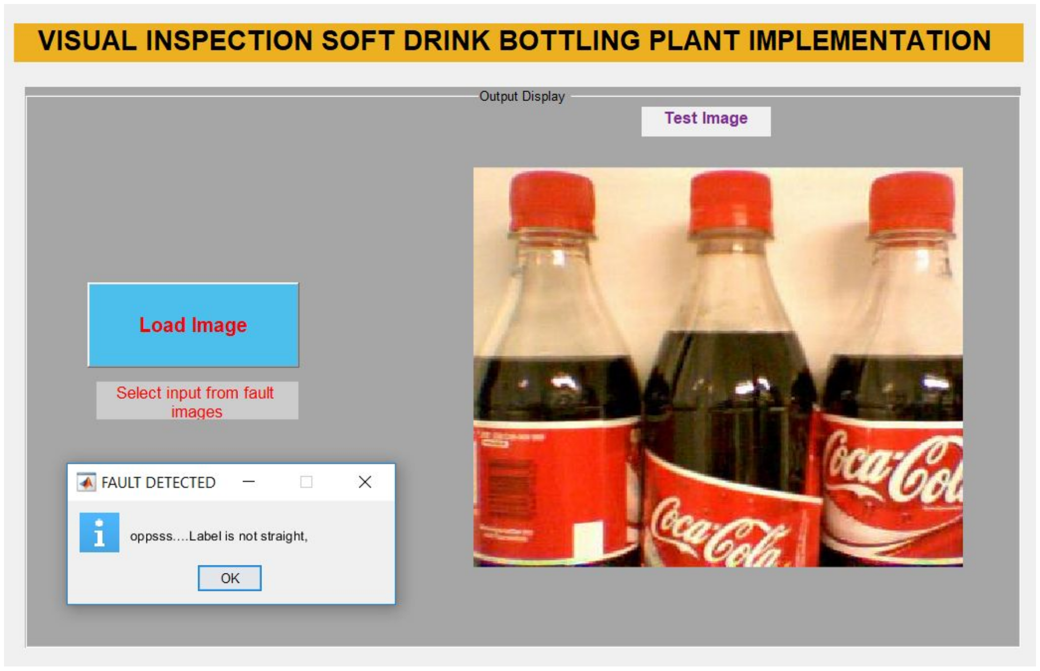
FAULT DETECTED

oppsss....! Label is not printed

OK

GUI Display for Label is not printed

6. Label is not straight:


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION

Output Display

Test Image

Load Image

Select input from fault images

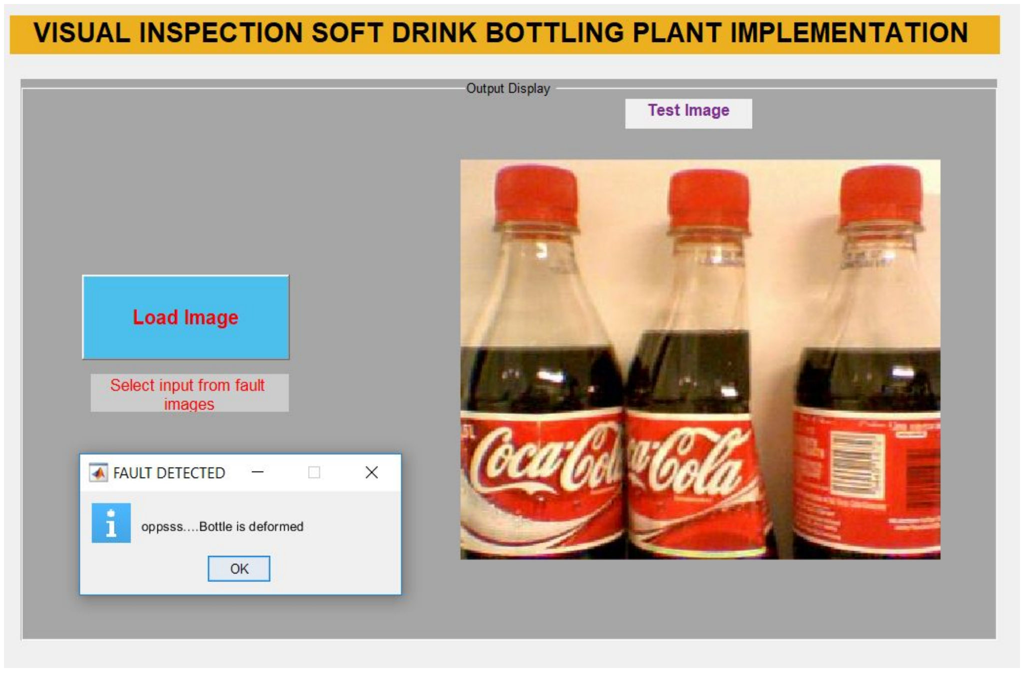
FAULT DETECTED

oppsss....Label is not straight,

OK

GUI Display for Label is not straight

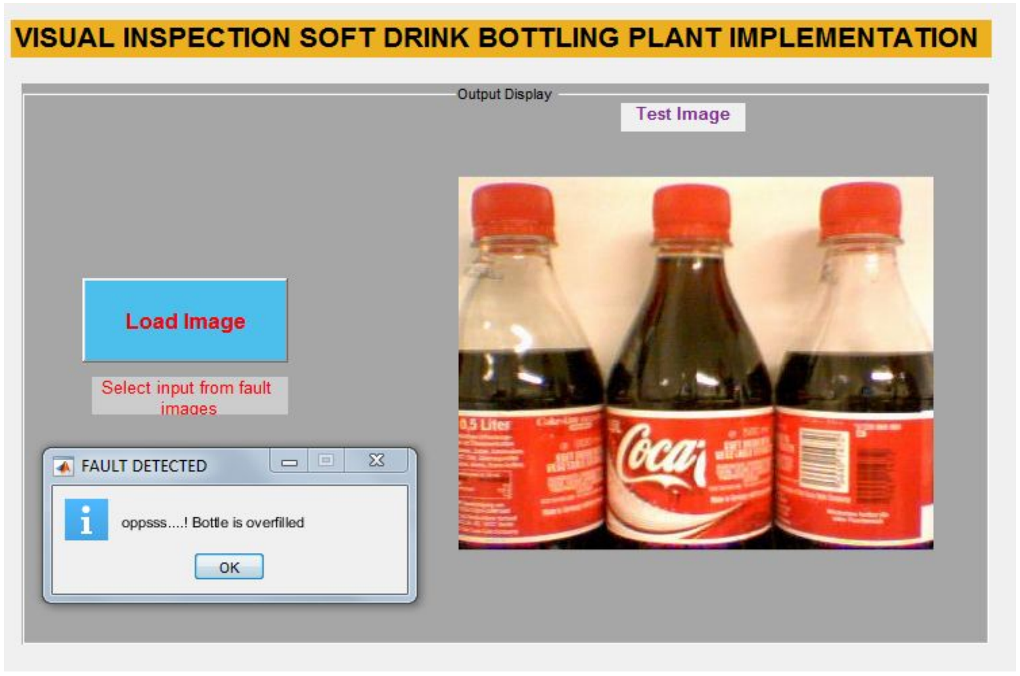
7.Bottle is deformed:


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
Load Image
Select input from fault images
FAULT DETECTED
OK

oppsss....Bottle is deformed

GUI Display for Bottle is Deformed

8. **Bottle is overfilled:**


VISUAL INSPECTION SOFT DRINK BOTTLING PLANT IMPLEMENTATION
Output Display
Test Image
Load Image
Select input from fault images
FAULT DETECTED
OK

oppsss....! Bottle is overfilled

GUI Display for Bottle is overfilled

Percentage Accuracy of the Algorithms:

Fault Type	**Images**	**Fault Detected**	**Classification**
Bottle Cap missing	9	9	**100 %**
Bottle Underfilled	9	9	**100 %**
Label Missing	9	9	**100 %**
Label not Printed	9	9	**100 %**
Label not Straight	9	9	**100 %**
Bottle Overfilled	9	9	**100 %**
Bottle Deformation	9	8	**90 %**
Bottle Missing	9	9	**100 %**

NOTE: We observe that for deformation bottles with softdrink inside will results in multiple faults. In Cap detection, we realized the bottles having overfilled liquid inside.

Conclusion:

We use Machine Vision techniques to find the faults in the industrial applications like this one in Soft Drink Bottle Plant. We realized the project by Matlab simulation. We arrived at the fault detections of the project/task assisted with accurate test results.

References links:

1. **https://pdfs.semanticscholar.org/a8a0/c11303ff5fe5d415a58b3ddeece3e1c877ae.pdf**
2. **https://fr.mathworks.com/help/matlab/ref/rgb2gray.html?searchHighlight=rgb2gray&s_tid=doc_srchtitle** **(We take all the required material for the function descriptions for using matlab inbult functions)**
3. **http://ictactjournals.in/paper/IJIVP_V6_I2_paper_3_1122_1126.pdf**
4. **https://github.com/Toemazz/ProductionLineVisualInspection**
5. **https://inspirit.net.in/books/academic/Computer%20and%20Machine%20Vision%204e%20-%20Theory,%20Algorithms,%20Practicalities%20By%20E%20R%20Davies.pdf**