Machine Vision - Automatic Inspection in Bottling Plant

ASRAF ALI Abdul Salam Rasmi
LUKAMBA Deogratias
MACAULAY Sadiq

Masters in Computer Vision
University of Burgundy
January 13, 2019

Supervisor: **Prof. SIDIBE Desire**
VIBOT, IUT Le Creusot, UMR6306 CNRS/uB

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

A machine vision system is a technology that enables a computing device to inspect, evaluate and identify moving or still images. It is a field in computer vision and is quite similar to surveillance cameras, but provides automatic image capturing, evaluation and processing capabilities. A machine vision system primarily enables a computer to recognize and evaluate images. It is similar to voice recognition technology, but uses images instead. A machine vision system typically consists of digital cameras and back-end image processing hardware and software. The camera at the front end captures images from the environment or from a focused object and then sends them to the processing system. Depending on the design or need of the Machine Vision System, the captured images are either stored or processed accordingly[1].

1.2 Objective

Our goal is to design a visual inspection system to correctly identify the following faults that can occur in a Soft Drink Bottling Plant.

- bottle under-filled or not filled at all
- bottle over-filled
- bottle has label missing
- bottle has label but label printing has failed (i.e. label is white)

- bottle label is not straight
- bottle cap is missing
- bottle is deformed in some way
- bottle missing

Chapter 2

Background

2.1 Binary Image

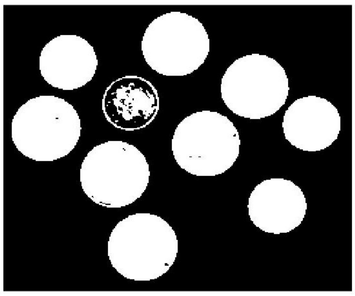
A ***Binary Image*** is an image in which each pixel assumes one of only two possible discrete (logical) values: 1 or 0. The logical value 1 is variously described as 'high', 'true' or 'on', whilst logical value 0 is described as 'low', 'false' or 'off'. In image processing, we refer to the pixels in a binary image having logical value 1 as the image foreground pixels, whilst those pixels having logical value 0 are called the image background. pixels[2].

A binary image is produced from a grey-scale or colour image by setting pixel values to 1 or 0 depending on whether they are above or below a threshold value. This process is called ***Thresholding*** [2]. This technique is commonly used to separate or segment a region or object within the image based upon its pixel values, as shown in Figure **??**. In its basic operation, thresholding operates on an image I as follows.

Algorithm 1: Thresholding Algorithm

```
    Input: Gray Scale Image
    Output: Binary Image
for I(i, j) ∈ I do
    if I(i,j) > Threshold then
        I(i,j) = 1;
    else
        I(i,j) = 0;
    end
end
```

(a) Gray Scale Image



(b) Binary Image

Figure 2.1: Thresholding for Object Identification

2.2 Binary Image Analysis

The goal of performing Binary Image Analysis is to find appropriate shapes in a binary image by grouping together the connected groups of pixels. This concept is called ***Region Labelling***. We can extract various properties of these regions such as Area, Height, Width, Centroid, etc. to detect objects and its morphology (say, defects). In order to perform this we check for Pixel connectivity which means to check for similar pixels in the neighborhood[3].

Chapter 3

Implementation

3.1 Methods Used

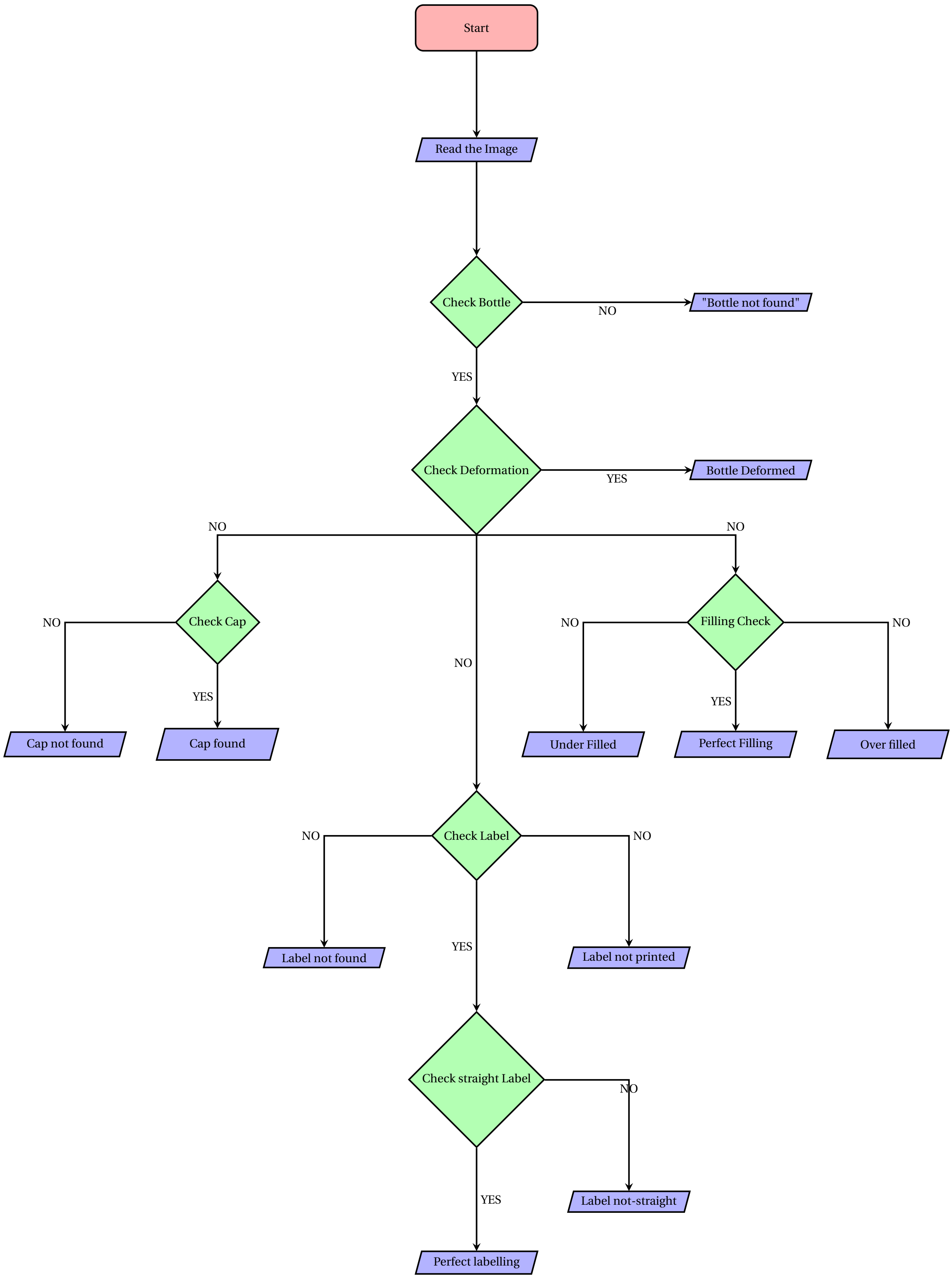
Binary Image Analysis is the principal concept used in this project to check for defects in bottling plant. Here we use MATLAB built-in function $imbinarize$ to convert the image into binary. To measure the area we use the function $regionprops$ which returns the area of all white regions in the image in a $struct$ format. Other than these functions morphological operations like $imopen$, $bwareaopen$, $imcomplement$ were also used.

3.2 Algorithm

The Proposed Algorithm takes an image as input and checks for the defects and displays the faults in five categories as follows.

- Check for Bottle.
- Check for Deformation.
- Cap Detection.
- Filling Check.
- Label Check.

Each category shows an unique result. A flow chart which briefly explains the working schema of our algorithm is given below.


Start
Read the Image
Check Bottle
NO
"Bottle not found"
YES
Check Deformation
YES
Bottle Deformed
NO
NO
Check Cap
NO
YES
Cap not found
Cap found
NO
Filling Check
NO
NO
YES
Under Filled
Perfect Filling
Over filled
Check Label
NO
NO
Label not found
YES
Label not printed
Check straight Label
NO
YES
Label not-straight
Perfect labelling

3.2.1 Check for Bottle

Algorithm 2: Check for Bottle

```
    Input: Image
    Output: Message
Crop the Image;
Filter the Image (imopen);
Binarize the Image;
if Region < Threshold then
    Remove the Region
else
    Calculate the White Region
end
if White Region < Area then
    Message: Bottle Found;
else
    Message: Bottle Not Found;
end
```

3.2.2 Check for Deformation

Algorithm 3: Check for Deformation

```
    Input: Image
    Output: Message
Crop the Image;
Binarize the Image;
Calculate the 'A' White Region;
Invert the Image;
Calculate the 'B' White Region;
Area = Max(A,B);
if Area > Threshold then
    Message: Normal Bottle;
else
    Message: Deformed Bottle;
end
```

3.2.3 Cap Detection

Algorithm 4: Cap Detection

Input: Image
Output: Message
Crop the Image;
Filter the Image ($imopen$);
Binarize the Image;
Invert the Image;
if *Region* < *Threshold* **then**
    Remove the Region
else
    Calculate the White Region
end
if *White Region* < *Area* **then**
    Message: **Cap Found;**
else
    Message: **Cap Missing;**
end

3.2.4 Filling Check

Algorithm 5: Filling Check

```
    Input: Image
    Output: Message
Crop the Image;
Filter the Image (imopen);
Binarize the Image;
Invert the Image;
if Region < Threshold then
    Remove the Region
else
    Calculate the Area of White Region
end
if Area > Threshold then
    Message: Overfilling;
else if Area < Threshold then
    Message: Underfilling;
else
    Message: Correctfilling;
end
```

3.2.5 Label Check

Algorithm 6: Label Check

Input: Image
Output: Message
Crop the Image;
Binarize the Image;
Filter the Image ($bwareaopen$);
Calculate the White Region;
Sum Area of All White Region;
if *Area* < *Threshold* **then**
    Message: **Label Missing;**
else if *Area* > *Threshold* **then**
    Message: **Label Not Printed;**
else
    Find the Max Area;
    if *Max Area* < $Threshold^1$ **then**
        Message: **Label Not Straight;**
    else
        Message: **Perfect Labelling;**
    end
end

Chapter 4

Results

We tested our Algorithm using MATLAB and it works with 99.9% Accuracy. The algorithm is designed in a way that if bottle is missing or bottle is deformed it does not check for other defects. The GUI layout and Output for various defects are shown in the following figures.

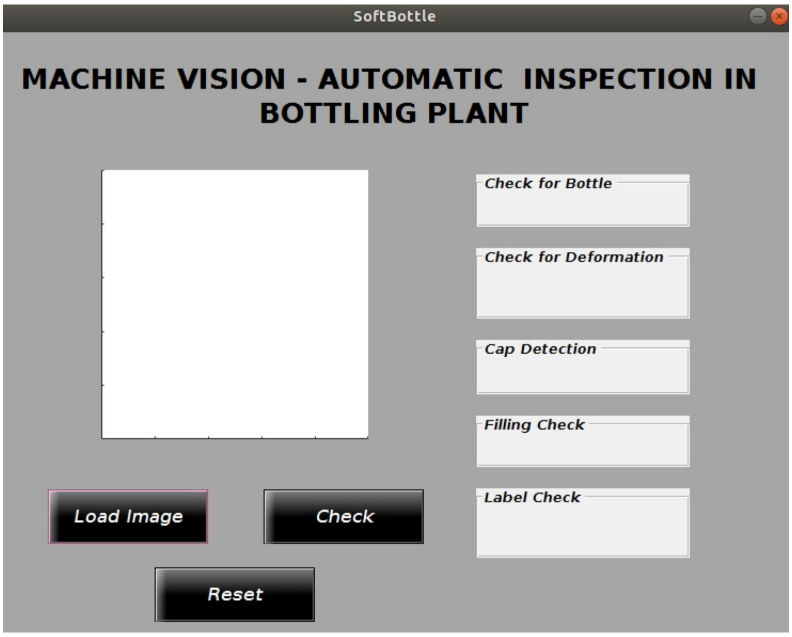


Figure 4.1: GUI Layout

Figure 4.2: Normal Bottle

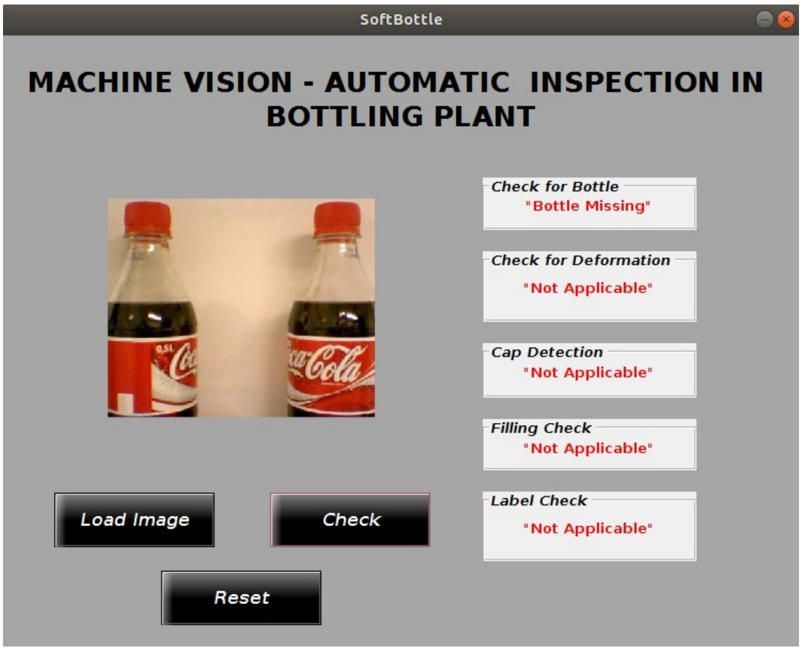


Figure 4.3: Bottle Missing

Figure 4.4: Deformed Bottle

Figure 4.5: Cap Missing / Overfilling

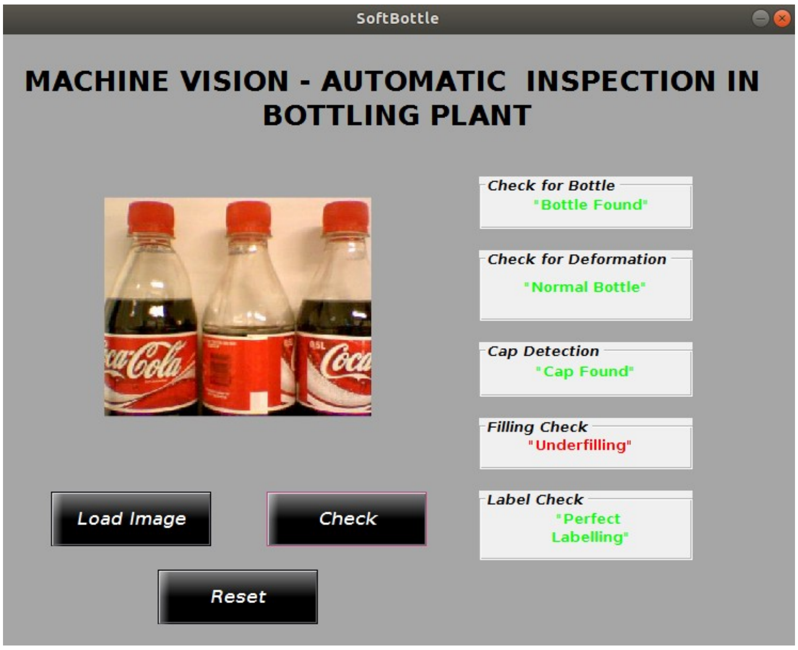


Figure 4.6: Underfilling

Figure 4.7: Label Missing

Figure 4.8: Label Not Printed

Figure 4.9: Label Not Straight

Chapter 5

Conclusion

5.1 Conclusion

The main objective of this project was to understand the basic concepts Image Processing. We applied only the basic image processing techniques to design a Machine Vision System for fault detection in Bottling Plant. This system can be further modified using Deep Learning Techniques for greater accuracy and Precision.

5.2 What we learned...?

1. We learned in detail and understood how to implement the concepts of Image Processing in Real Life.
2. The techniques of Binarization and Morphological Processing are clear and applicable in further aspects.
3. We learned MATLAB-gui and some in-built functions like 'imbinarize' (to convert rgb or gray imag to binary), 'regionprops' (to compute area and other parameters of a region), etc.
4. We developed our programming skills in MATLAB.

Bibliography

[1] "What is a machine vision system (mvs)? - definition from techopedia." `https://www.techopedia.com/definition/30414/machine-vision-system-mvs`. Accessed: 2019-01-09.

[2] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab.* Wiley-Blackwell, 2010.

[3] D. Sidibe, "Introduction to image processing," in *Lecture Notes*, 2018.