HW10_VYAS_KHYATI (kv1060)

Production Line Visual Inspection: Coca-Cola Bottling Plant

Prof.Kinsman mentioned that no single model can be used to classify an object.

Therefore, I followed the process of elimination to find out if the bottle was defective or no. I created different models to find defects in the bottle moving along the chain to finally find if the bottle was deformed or no.

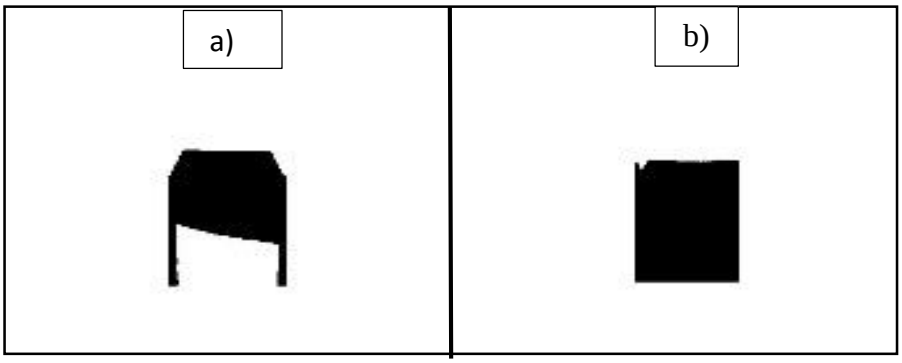
After, looking at the images, I realized that the approximate location of the center bottle is nearly the same for all the images. The following is my imaging chain:

IMAGING CHAIN:

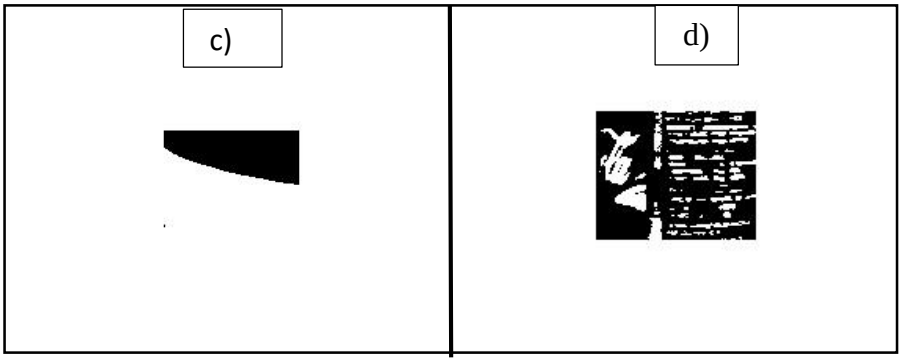
1. **Read the file**: The file is taken as input.
2. **Noise removal**: The noise that may be present in the image, is taken out using a Gaussian filter.
3. **Empty Space**: After reading the input, I crop the part of the image that contains only the central bottle.
    a. I then convert it into binary. This leads to 2 possibilities.
        i. If the area is empty i.e. no bottle is present, the binary image will be completely white.
        ii. If the area does have a bottle, the binary image will contain some black regions too.
    b. Now I calculate the number of connected components. An image with empty space will have 1 connected component. Whereas, an image with a bottle present will have more than 1 connected component
    c.
4. **Cap checker:**
    a. I wrote a function capChecker() to test if the cap is present or not.
    b. I calculated the approximate location of the cap in the image and cropped that part.
    c. Then for every pixel, I calculated the R,G,B values using impixel().
    d. Using the data cursor with various images, I found that the G and B pixel values change drastically when the cap is present and when it is not present. Thus I set a threshold on these values, to decide if the cap was present or not.

e. **Degree of confidence metric:** Threshold on the G and B values.

5. **<u>Label Detector</u>**:
   a. This consists of 3 parts:
      i. Check if the label is present at all? Images of bottles with no labels are rejected.
      ii. Check if the label is printed or not? Images of bottles with white labels are rejected.
      iii. Check if the label is properly placed. Images of bottles with crooked labels are rejected.
   b. I wrote a function called labelDetector().
      i. I cropped the part of the image which contains the bottle. Since the label mainly consists of red, I extracted the red plane and converted the image into a binary image.
      ii. Now the image consists of the parts of the bottle which show the filled cola in black and the cap, label in white.
      iii. Thus for a bottle with no label, will give me a completely black image for the lower region and completely white for the upper region. (b)
      iv. And, a bottle with a label will give a lower black region but with the label in white.(a)
      v. Now I calculated the number of connected components. Bottles with no label will have only one connected components. Whereas, bottles with labels will have more than one connected component.
      vi. **Degree of confidence metric**: Number of connected components.
      vii. Based on this, the bottle is rejected if it contains only 1 connected component.



c. Next to check if the label is printed properly or not. Images containing white (non-printed) labels need to be rejected. I wrote a function called whiteLabelDetector().
    i. I cropped the part of the image that contained only the label. Now since the label contains majorly red, I used the green and the blue planes.
    ii. Now following the same procedure as above, I converted the image to binary and calculated the number of connected components. The bottles with white labels will give me 1 connected component if the label is positioned correctely (i.e. not crooked) or less than 3 connected components, if the label is slanted (c).
    iii. The labels with print on them will contain many connected components, due to the text (d).
    iv. **Degree of confidence metric**: Number of connected components.



d. Next to check if the label is slanted or placed properly, I wrote a function called slantedLabelDetector();
    i. For this I created a template image, which is an image of an idealistic bottle. I considered (image103.jpg) to create this template
    ii. I also cropped the part of the image that contains the center bottle. And converted to binary using the red plane alone.
    iii. I converted the template to binary using the red plane.
    iv. I used morphological cleaning for removing unwanted regions from both images.
    v. Then I performed template matching using normxcorr2(). This gives me a value for the cross correlation between each pixel in both the images.
    vi. I found out the maximum of these values, and based on this max_values, classified if the bottle had a slanted label or no.
    vii. If the max of cross correlation value was less than 0.7 then it has a slanted label.
    viii. **Degree of confidence metric**: Maximum of cross correlation values.

6. **Level Detector**:
    a. I wrote a function to detect if the level of the cola was higher /lower or the normal amount.
    b. For this I cropped part of the image which partially covers the area where the cola is filled. i.e. Just above the label.
    c. Now based on this cropped image, if the entire bottle is full, or if the bottle is over filled, the binary of this image will be completely black.
    d. If the bottle is filled normally at the right level, the binary of this image will contain one half black and the upper half white.
    e. If the bottle is completely empty, then the entire image will be white. To detect this as an empty bottle, I took the complement of the binary image and thus, it will be completely black.
    f. Then I calculated the number of connected objects.
    g. Pseudocode:

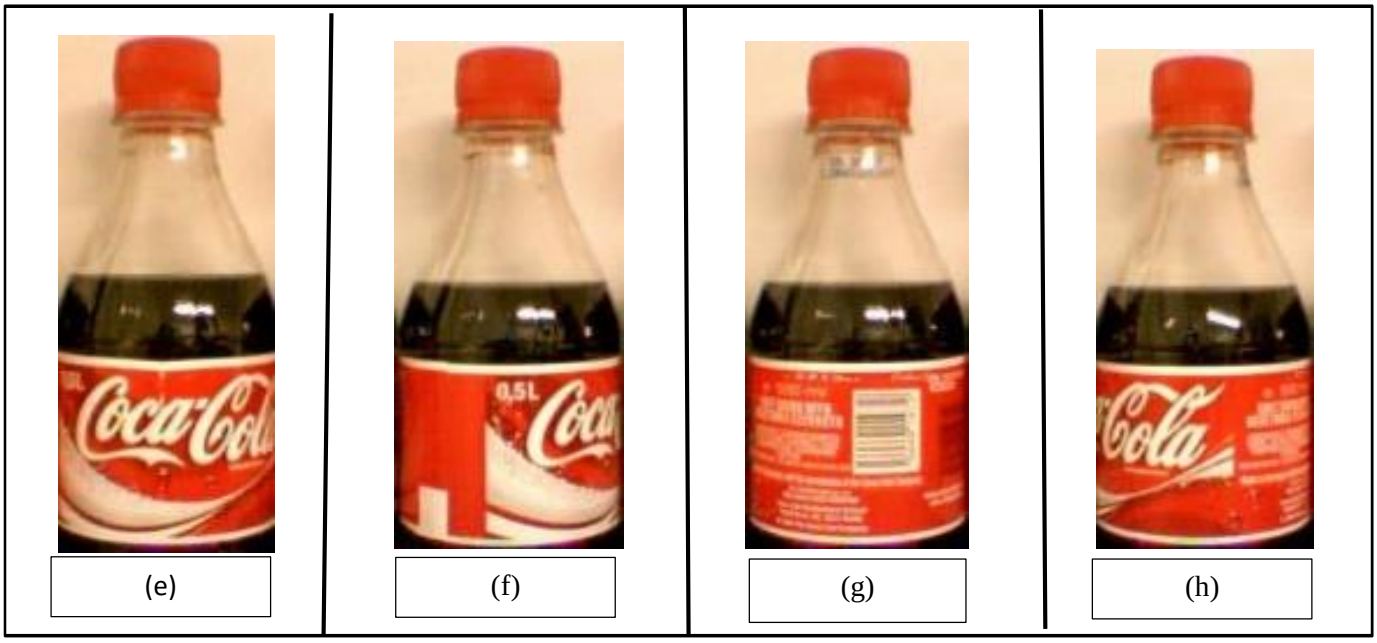
```
If (number of objects == 0)
        {'BOTTLE IS OVERFILLED. REJECT BOTTLE!!'}

Imcomplement (binary_image);
        If (number of objects == 0)
                {'BOTTLE IS UNDERFILLED. REJECT BOTTLE!!'}
        Else
                {'LEVEL IS FINE! GO AHEAD'}
        End
End
```

    h. **Degree of confidence metric**: Number of connected components.

7. **Deformed Bottle:**
    a. I wrote a function crushedBottle() to detect if the bottle was deformed or not.

b. For this I took 4 different samples of ideal bottles which consisted of bottles with changed orientation of labels.
    i. Coco-cola text exactly in the center. (e)
    ii. Coco-cola text more towards the right. (f)
    iii. Coco-cola text not seen at all. (g)
    iv. Coco-cola text more towards the left. (h)



(e) (f) (g) (h)

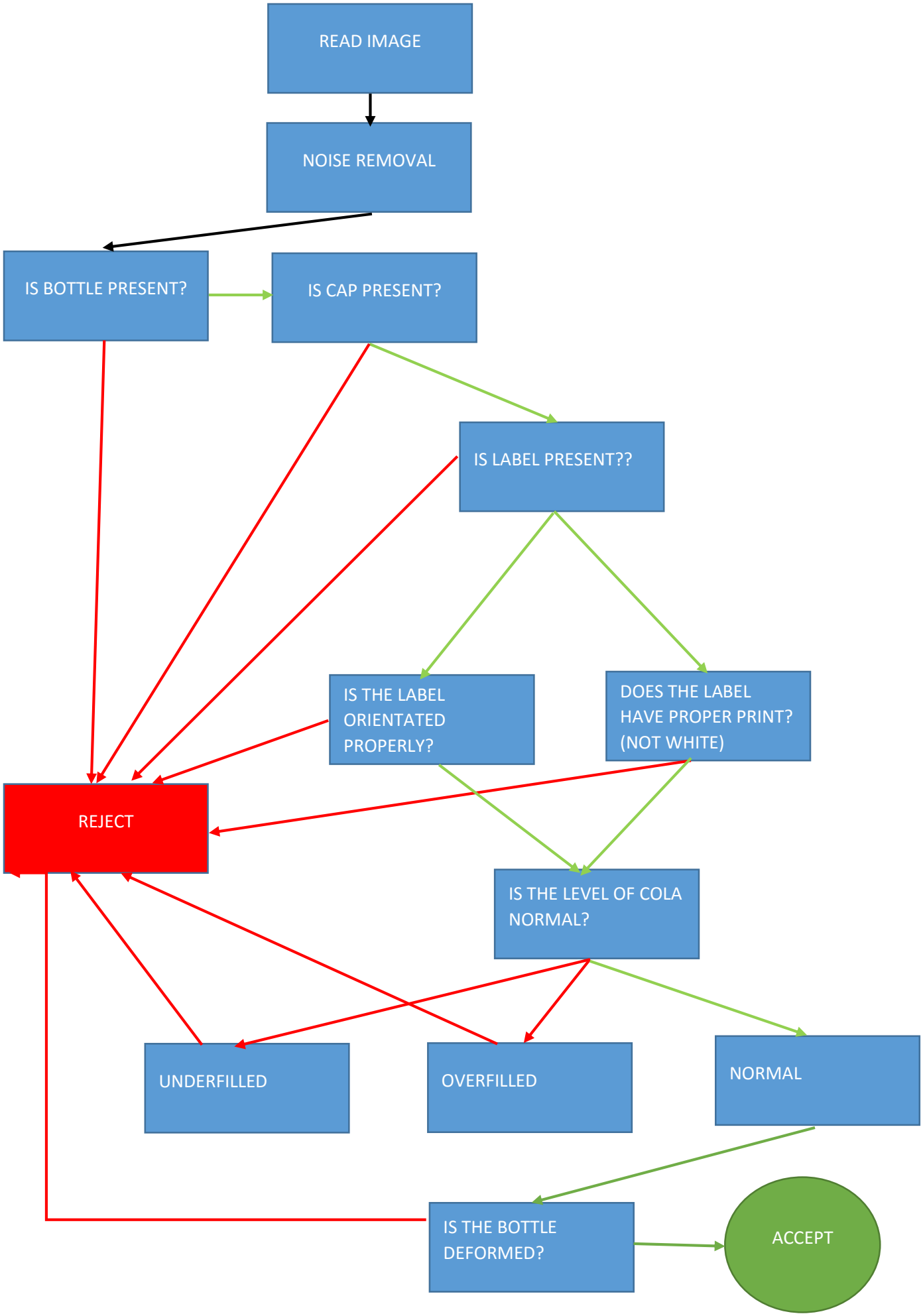
c. Now I performed template matching on the input image with all the 4 template models.
d. And calculated the max cross correlation value for each and based on the 4 value, I found the maximum value. If the max value of the 4 values was greater than 0.65, then the bottle was classified as not deformed.
e. **Degree of confidence metric**: Maximum of cross correlation values.

Conclusion:

The imaging chain is displayed diagrammatically below.

Red lines indicate NO.

Green lines indicate YES.


READ IMAGE
NOISE REMOVAL
IS BOTTLE PRESENT?
IS CAP PRESENT?
IS LABEL PRESENT??
IS THE LABEL ORIENTATED PROPERLY?
DOES THE LABEL HAVE PROPER PRINT? (NOT WHITE)
REJECT
IS THE LEVEL OF COLA NORMAL?
UNDERFILLED
OVERFILLED
NORMAL
IS THE BOTTLE DEFORMED?
ACCEPT