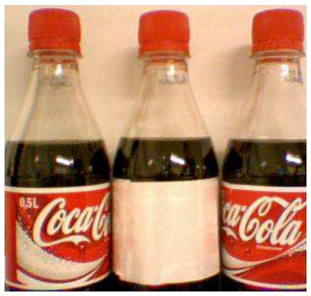
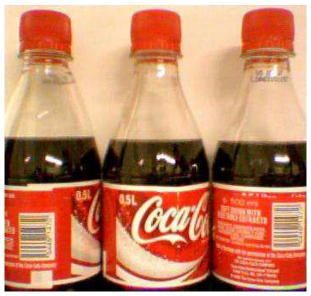
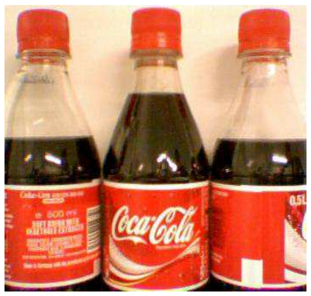
EE551 – Assignment 1 – Visual Inspection – V4, February 2022

Production Line Visual Inspection: Coca-Cola Bottling Plant

Background

A common use of image processing in an industrial setting is for the automated visual inspection of products leaving a production facility. Automated inspection is used to inspect everything from pharmaceutical drugs to textile production. This helps to reduce costs and reduce the likelihood of faulty products being shipped (manual inspection can be error prone).

In this practical exercise, we are dealing with a bottling production line in a facility bottling Coca-Cola for the Irish domestic market. We have a set of images, taken under near constant factory lighting conditions, of the bottles as they leave the bottling line. The bottling company require a vision system to automatically identify a number of specific faults that may occur during the filling, labelling and capping stages of production so that these bottles can be intercepted prior to packaging.

Your task is to design and prototype a vision system to detect the set of fault conditions that may occur together with identifying the type of fault that has occurred. You will develop this prototype system using the software and the techniques discussed in the course.

Task Specification – Automated Visual Inspection

You are required to develop a visual inspection system that correctly identifies each of the following fault conditions that may occur in the bottling plant:

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*** *(i.e. squashed) in some way*

The system is also expected to detect "normal" bottles, i.e. it should be able to recognise when a bottle has no faults at all ("true negative" cases).

In each image we are ***only 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. *Images with faults in the bottles on either side, or a missing bottle in the centre should be ignored by your system – only the seven faults above must be reported.* You can tackle each of the cases in any order you wish. Some examples contain more than one fault with the central bottle – identify as many as you can.

Test Data

The sample data provided is a set of images of bottles leaving the bottling facility. They have been additionally split into labelled sets for ease of prototype development. The images have been captured by a high-speed image camera as they leave the facility – although no motion blur occurs in the images, the positions of the bottles in the image varies slightly from image to image.

The sets of images are stored in Zip files in the folder "Image Zip Archives" under "Assignment 1" under "Assessment" on the EE551 Blackboard web page. These consist of: (i) Zip files containing labelled examples for each of the 7 faults; (ii) Additional Zip files with "normal", "bottle missing" and "combination of faults" images; (iii) a single Zip file with all of the image files randomised – this can be used for final testing of your system.

Submission

You must submit the following:

- Working Matlab .m script for implementing the system.
- Report detailing your approach to the problem and the success of your system in identifying each of the 7 bottling fault cases as well as handling combinations of faults, normal, side fault and missing bottle images. Provide any illustrative images (as many as you feel necessary) of the intermediate results of the system you produc*e (overlays, results of processing stages etc.).* Summarise the success of your system in identifying each fault case e.g. through giving the "percentage detection rate" for each fault, as well as an overall detection rate. As well as overall detection rates, you may also go into more detail in your analysis, e.g. how often does it detect a fault when none is present ("false positives") etc.
  A critique of your system is also particularly valuable i.e. where does your system work well? Where does your system fail, and do you know why (and if you had unlimited time, what would you do to improve it)?

Reports should be as long as they need to be and no longer (i.e. no "padding"!). Submit your report as a PDF or a Microsoft Word document. Further details on submission (incl. deadline) will be communicated to you via e-mail.

Notes

1. "System failures" primarily come in two forms (i) faults not identified or incorrectly identified; (ii) faults declared with "normal" (no fault) bottles ("false positives").
2. Marks are awarded for the quality and performance of your algorithm as well as for "working code", for each fault case. Your reports should include a description of your algorithm that is, in a sense, "software independent", so use flow charts, pseudo-code, equations etc. as you need to. In a way, the functionality is the most important part, and the Matlab implementation is simply a tool by which you demonstrate your algorithm in operation. Marks will also be awarded for the "critique"

of your system and its performance, i.e. if your system fails in certain situations, can you explain why?
3. Your program must run on a PC with MATLAB and a copy of the Image Processing toolbox – no third party tools or scripts may be installed.
4. You should structure your source code so that all that needs to be done is to "point" the software at a directory that contains a set of test images to be processed – these could be directories that contain only one type of fault, or it could be a directory that contains all of the randomised images. It **must** be possible to run your software with minimal editing i.e. only directory names should need to be changed to reflect differences in the file structure on your own laptop and those who will be assessing your submissions. For submission, you can have all of your code ("main" program and any functions) in one source M-file; however, if you do this, remember that the functions will have limited scope and will not be visible outside this source file.
Also, for submission, if TurnItIn has difficulties with the .m file format (it also can have difficulties with Zip files) you can improvise and save your Matlab script as a .TXT document or MS Word document and upload that – as long as it's uploaded in some "text" format, that's fine. You should also e-mail your report and source code (.m file) to brian.deegan82@gmail.com as well.
5. Tackle one fault condition at a time ("divide and conquer"), and write your code in such a way that each fault (or other condition) is processed by a Matlab function. This will make your code much easier to test and debug.

Marks

The marks for this practical will be awarded as follows:

- Correct identification of each of the fault conditions:
    - the 7 individual faults — 55%
    - combination of faults — 10%

  including ignoring missing bottles, images with "side faults" etc. by correctly reporting no fault
- Report: — 35%
    - Discussion / detail of system design and choices made
    - Evidence of the success of system in performing the specified task
    - Critical analysis of the results, suggestions for improvements etc.
    - General presentation quality of report

Total 100%

<u>Plagiarism</u>
You will no doubt be aware that the University treats plagiarism very seriously. Students are expected to work independently and to submit their own original report, without copying material from another person, or from another source. Submissions will be processed by Turnitin to detect plagiarism. Further information relating to plagiarism may be found at: http://www.nuigalway.ie/plagiarism/

Also, while you may discuss and debate elements of the assignment with your classmates, **each student must write their own code, and complete and submit their report independently of all other students** (TurnItIn will be used to check for plagiarism) – in other words, you can "collaborate, but don't copy".