Carnegie Mellon University Africa

04-800-G Applied Computer Vision

Assignment 2
Classifying Sweets

Deadline: 18:00 Thursday 26th October 2017

Problem Definition
The objective of this assignment is to develop a computer vision application that can analyse a scene, classify disk-shaped coloured sweets as normal or defective, and count the number of defective sweets (see Figs. 1-4). A defective sweet is defined to be one that has a piece missing from its outside edge (see Figs. 2 and 4). The residue of any defect (i.e. the missing piece) will not appear in the scene. In all cases, the scene background will be white and the sweets do not touch one another. You can assume that the defect, i.e. the missing part, appears as a concavity in the sweet boundary. The image will only contain sweets.

Your program should compute the following.

1. The total number of defective sweets.
2. The total number of colours
3. The number of defective sweets of each colour

For each test image, 50% of the marks will be allocated for the total number of defective sweets, 20% for the number of distinct colours, 20% for the number of defective sweets of each colour, and 10% for the coverage of the test data set.

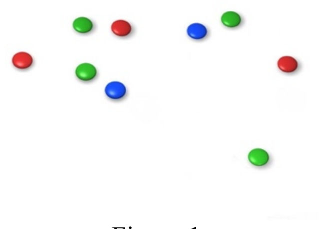
Figure 1

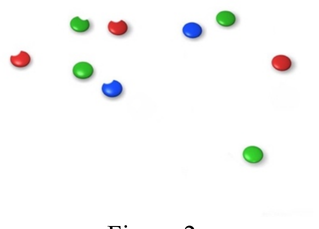
Figure 2

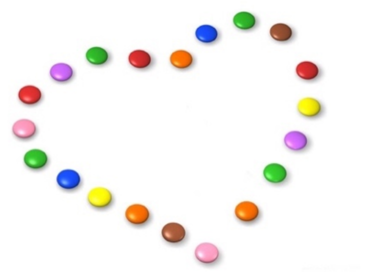
Figure 3

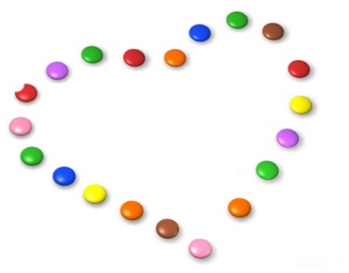
Figure 4

The application should read image input from image files, each file comprising an image of a scene comprising coloured sweets. The image file names should be provided in an input file named input.txt. The application should write the required data, i.e., the number of sweets in each image, to an output file named output.txt.

Input
The filenames of the images of the scenes containing coloured sweets.

Output
The output should begin with your Andrew Id on a separate line; your Id comprises the first letter of your first name and your last name in full (e.g. dvernon).

This should be followed, for each test case, by the number of defective sweets in the image, the number of colours, and the number of defective sweets for each colour. There should be one filename and set of numbers per line. If there are no defective sweets, report a zero. The number of defective sweets of each colour should be listed in increasing hue value.

Sample Input

```
../data/assignment2A.jpg
../data/assignment2B.jpg
../data/assignment2C.jpg
../data/assignment2D.jpg
```

Sample Output

```
dvernon
../data/assignment2A.jpg 0 defective sweets, 3 colours: 0 0 0 defects per colour
../data/assignment2B.jpg 4 defective sweets, 3 colours: 2 1 1 defects per colour
../data/assignment2C.jpg 0 defective sweets, 8 colours: 0 0 0 0 0 0 0 0 defects per colour
../data/assignment2D.jpg 1 defective sweets, 8 colours: 1 0 0 0 0 0 0 0 defects per colour
```

Instructions
Submit the following in a zip file named with your Andrew Id to vernon@cmu.edu by the deadline shown above.

1. The source code: *.c, *.cpp, and *.h file(s)
2. The cmake file: CmakeLists.txt
3. The test input file: input.txt
4. The test output file: output.txt
5. The image files you used to test your program.

Do not include the Visual C++ solution files or the executable. Submit only the source code files, the Cmake file, and the input, output, and image files; do so in a single compressed (zipped) folder. *Do not include subdirectories in the zipped folder*.

The source code should contain adequate internal documentation in the form of comments. Internal documentation should include the following.

- A summary of the algorithm
- A summary of the testing strategy.

Do not forget to include your name in the internal documentation. Place this documentation in the .h include file.

For this assignment, no marks will be awarded for internal documentation but it is good practice to include it.