COMP 9517

T3, 2020

Assignment 1: Specification

Maximum marks achievable: 10 marks

This assignment is worth 10% of the total course marks.

The assignment files should be submitted online.
Deadline for submission is Week 4 Monday 5 October 2020, 23:59:59 AEDT
Instructions for submission will be posted closer to the deadline.

Rice is a staple food for a large part of the human population. Commercially the grading of rice is performed based on several factors such as weight, purity, percentage of damaged kernels and presence of foreign material. In this assignment you will explore the use of Image Processing techniques to grade rice based on the percentage of damaged kernels.

Objectives: This assignment is aimed at familiarization with basic image processing methods. It also introduces you to common image processing and analysis tasks using OpenCV. After completing this assignment, you will learn how to:

1. Open and read image files
2. Perform simple mathematical operations on images
3. Perform segmentation using thresholding
4. Assign labels to connected components

Deliverables: You will submit a report (maximum 5 pages) briefly explaining the approach you have taken in Tasks 1, 2 and 3 and include some sample input images and the intermediate and final images obtained. You must also submit the Python source code files.

Submission: The assignment files should be submitted online via WebCMS. Instructions for submission will be posted closer to the deadline.

Tip: You are advised to use OpenCV 3+ with python 3+.

Task 1: Iso-data Intensity Thresholding (4 marks)

Iso-data intensity thresholding is a simple method to determine a threshold to separate pixels in a greyscale image. It assumes that the image contains two classes of pixels - *foreground* and *background*- and has a *bi-modal pixel intensity histogram*. It then attempts to find an optimal threshold value, to divide the image into foreground and background. The algorithm is explained below.

```
1. Select an arbitrary initial threshold t
```

2. Compute μ_0 mean of intensity values < t and μ_1, mean of intensity values > = t
3. Update the threshold to the mean of the means $t_{new} = (\mu_0 + \mu_1)/2$
4. If the threshold has changed non-trivially in step 3 go to step 2. (Hint: To check whether the change in threshold value is non-trivial, test the difference between old and new threshold values against a small epsilon value, for example 0.01. You should experiment with a few different epsilon values before choosing one.)

Upon convergence, the threshold is midway between the two class means.

Implement the Iso-data intensity thresholding method on the images provided, to find a good threshold value for each image. Then generate a binary image with the background encoded as white [pixel value: 255] and the rice kernels as black [pixel value: 0]. Finally, plot the threshold value t at every iteration on a graph.

- Input: greyscale image
- Output: binary image, threshold value, plot of threshold value t at every iteration on a graph

Task 2: Counting the rice kernels (4 marks)

In this task you are required to count the number of rice kernels in each image by counting the number of connected components in the image.

As a pre-processing step, noisy pixels in the binary image generated in Task 1 should first be removed. Apply a median filter to the thresholded binary image produced in Task 1, to remove any noisy pixels.

Then using the two-pass connected components algorithm (explained below), iterate through the binary image. During the first pass temporary labels are assigned to each data point and the equivalences are stored, and during the second pass the temporary labels are replaced by the smallest label from its equivalence class. Use the connected component labels assigned in the second pass to isolate the rice kernels in the original image, and then count the number of rice kernels. Consider two pixels to be adjacent if they share the same colour / intensity and each belongs to the other's eight-neighbourhood, as illustrated below:

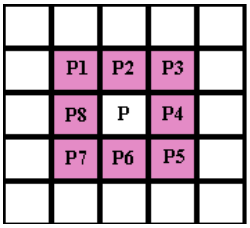

Figure 1: 8-neighbourhood (P: centre pixel; P1-P8 neighbouring pixels)

You should implement the two-pass connected component labelling algorithm as described below:

First pass:

```
Iterate through each pixel in the image matrix:
      If the pixel is not the background:
            Get the neighbouring elements of the current pixel
            If there are no neighbours:
                  uniquely label the current pixel and continue
            else:
                  find the neighbour with the smallest label and assign
                  the same label to the current pixel
            Store the equivalence between neighbouring labels
```

Second pass:

```
Iterate through each pixel of the data by column, then by row
      If the pixel is not in the background:
            Relabel the element with the lowest equivalent label
```

(Source : https://en.wikipedia.org/wiki/Connected-component_labeling)

- Input: binary image generated in Task 1
- Output: filtered binary image, number of rice kernels in the image

Task 3: Percentage of damaged rice kernels (2 marks)

Damaged rice kernels occupy a smaller area in comparison to the whole kernels. Given a minimum pixel area 'min_area', using the connected component labels generated in Task 2, find the number of damaged kernels (components occupying a pixel area < min_area) and the number of whole kernels (components occupying a pixel area > = min_area). Divide the number of damaged kernels by the total number of rice kernels in the image to get the percentage of damaged rice kernels.

- Input: connected components labelled from Task 2, threshold value 'min_area' (choose an appropriate value which can separate the damaged kernels from the whole ones)
- Output: percentage of damaged rice kernels, binary image excluding all the damaged kernels

ORGANIZING THE CODE

The assignment code should be submitted as a python file named ***Assignment1.py***. Each task should be written as separate functions within this file and should be executable from a command line. The input details should be passed to the file as command line arguments (outline code provided). The input image should be readable from the location specified as an

argument, and all output images should be saved in a new sub-folder in the code path (see code organization below). The output images of each task should be named appropriately. In addition to an image, an output value is generated during each task, and this value should be written as a title while saving the image (example shown below).

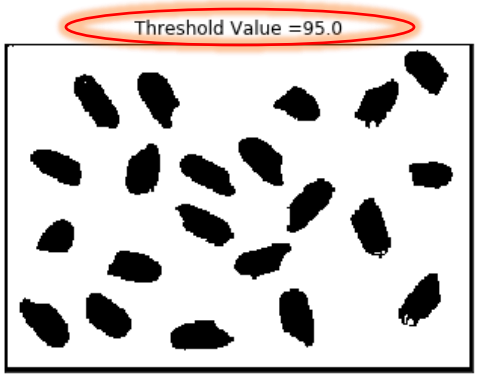

Figure 2: rice1_Task1.png

Sample images and outline python code can be downloaded from the course website. The code should be organized as shown below (| | stands for folder):

| code |

Assignment1.py

Input images may reside here or any other location

| OUTPUT |

*Input_filename*_Task1.png

*Input_filename*_Task2.png

*Input_filename*_Task3.png

DONOT use library function from OpenCV (or any other packages) such as

`threshold`
`adaptiveThreshold`
`watershed`
`findContours`
`contourArea`
`drawContours`
`connectedComponents`
`medianBlur`
`SimpleBlobDetector`

or similar ones.

18 September 2020