**Summative Assignment 2016**

**Module:** Image Processing (COMP2231)

**Lecturer:** Boguslaw Obara

**Deadline:** 14:00, 20/1/17

**Submission: C**ode and report should be submitted via DUO

**Problem:**

Given in a selection of images are controls selected from a screen to find novel anti-infectives using the roundworm *C.elegans*. The animals were exposed to the pathogen *Enterococcus faecalis* and either untreated or treated with ampicillin, a known antibiotic against the pathogen. The untreated (negative control) worms display predominantly the "dead" pheotype: worms appear rod-like in shape and slightly uneven in texture. The treated (ampicillin, positive control) worms display predominantly the "live" phenotype: worms appear curved in shape and smooth in texture. For more information, please see: [Moy et al. (*ACS Chem Biol*, 2009)](http://dx.doi.org/10.1021/cb900084v).

**Data:**

*Images:*

* One image per channel (Channel 1 = brightfield; channel 2 = GFP) was acquired at MGH on a Discovery-1 automated microscope (Molecular Devices). Original image size is 696 x 520 pixels. Images are available in 16-bit TIF.

BBBC010\_v1\_images.zip (64 MB)

*Ground truth:*

* The images are from a plate of positive and negative controls. The images are named using this format: <plate>\_<wellrow>\_<wellcolumn>\_<wavelength>\_<fileid>.tif Columns 1-12 are positive controls treated with ampicillin. Columns 13-24 are untreated negative controls.
* Human-corrected binary images of foreground/background segmentation: BBBC010\_v1\_foreground.zip (1.1 MB)
* To address the problem of correctly segmenting individual worms also when they overlap or cluster, we provide one binary foreground/background segmentation ground truth image for each worm: BBBC010\_v1\_foreground\_eachworm.zip (2.7 MB)

**Task:**

This exercise asks you to write a Python program to understand the principles of image segmentation concept. You may use another language if you prefer, but the same principles of readable and reusable code applies.

Each function should include:

* A good help text explaining the function usage, found by the help command. Try using headers!
* Enough comments to make the user in understanding the algorithm.

Try to keep all your functions simple. They should be re-used by other functions making the code easier to read and keeping it organized.

**Marking:**

The marks for this assessment coursework will be awarded as follows:

* The quality, readability and usability of their code (10%).
* Worms detection algorithm:
  + DONEReading both image channels (5%)
  + DONEImage segmentation (15%)
  + DONEWorms detection: background separation, objects labeling and counting (20%)
  + DONE SORT OFDetection of individual worms when they overlap or cluster (10%)
  + Live/dead warm classification using shape and texture information (10%)
* Evaluation
  + DONEComparison of your warms detection results with the *Ground truth* data (10%)
* DONE YOU KNOW IT, ALWAYSClear, well documented program source code (10%)
* Report:
  + Discussion / detail of system design and choices made (5%)
  + Evidence of the success of system in performing the specified task (5%)

**Submission:**

You must submit the following:

* Full program **source code** for your solution to the above task.
* **Working Python script** meeting the above specifications.
* **Report (max. 750 words!)** detailing your approach to the problem and the success of your system in identifying each of the fault cases as well as handling no fault and fault combination images. Provide illustrative images (as many as you feel necessary) of the intermediate results of your system *(overlays, results of processing stages etc.).*

Make it clear in the initial comments of your source code how to run your script. Your script must run on one of the teaching lab based PCs – ensure compatibility before submission. If your solution relies on one or more “reference images” make sure these are included with your submission and make it clear where they need to be placed to ensure correct operation.

**Plagiarism:**

You must not plagiarise your work. You may use program source code from the provided course examples, the OpenCV library itself or any other source BUT this usage must be acknowledged in the comments of your submitted file. Automated software tools will be used to detect cases of source code plagiarism in this assignment exercise. This will include automatic comparison against code from previous year groups but will also take account of common code examples given out as part of the course (for the avoidance of doubt, we use the following plagiarism detection tool: <http://theory.stanford.edu/~aiken/moss/>). Attempts to hide plagiarism by simply changing comments/variable names will be detected. Plagiarism is not fair on those who work hard on this assignment without resorting to plagiarising the work of others. The School will always seek to address any instances of plagiarism following the rules set down by the University.

You should have been made aware of the Durham University policy on plagiarism. Anyone unclear on this must consult the course lecturer prior to submission of this assignment.

**To submit your work create a directory named by your username (e.g. *abs123*). Place all required files in this directory. ZIP (not rar or .z7 etc.) this entire directory structure and submit it via DUO *(late submissions will be penalised following school policy).***