Discussion/detail of system design and choices made (5%)

System Design

I wrote this script using Python 3.5.2, and OpenCv 3.1.0, to match the versions available on the DUDE PCs in the School. Operation of the script is done via key presses, and output of the program is both visual in the main window, as well as various images being saved to disk, in the img_out/ directory. The main window displays one file ID at a time, with the top row being the w1 channel, and the bottom being w2. These are labelled, and labelling can be toggled by pressing the l key. Full list of commands:

- n - next image
- b - previous image
- l - toggle labels
- s - save the currently displayed w1 and w2 processed images
- x - exit

There are also boolean settings toward the top of the script, to allow for automatic saving of various images.
Processing of images is done in steps, with each step being a standalone method. All methods are documented and commented. The processing steps are as follows:

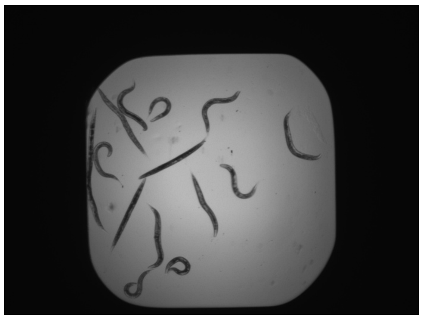
1. Isolate and Compare
   (a) **Isolate** the worms from the background and border.
   (b) **Compare** the isolated worms against the provided ground truth.
2. Individualise and Save each worm
   (a) Run the **watershed** algorithm to mark worms with unique colours.
   (b) Save each individual worm to disk, under img_out/separated/ directory.
3. Label each worm either dead or alive
   (a) Find the rotated bound box of each contour, and classify dead or alive depending on the width/height ratio of said box.

Examples of what the above steps produce can be found under later sections. All images shown here are of A01, to clearly show the changes.

Choices Made

As the images are saved as 16 bit but only really using 12 bits, when reading as 8 bit images the 4 most significant bits were lost, and the image was very dark (Figure 1), and so I read in the images as 16 bit, right shifted by 4 bits, and then converted to 8 bit. This ignores the 4 least significant bits, in favour of the 4 most significant bits. This results in a much more usable and bright image (Figure 2).

(a) Loaded as 8 bit


(b) Loaded as 16 bit and shifted

Evidence of the success of system in performing the specified task (5%)

Processing Steps

tqvj24