Discussion and detail of system design and choices made

System Design

I used Python 3.5.2, and OpenCv 3.1.0, matching the versions available on the DUDE PCs in the School. Key presses control the script, and output is shown in the main window, as well as various images being saved to disk, in the img_out/ directory. One file ID is displayed at a time, the top row being the w1 channel, and the bottom being w2. These are labelled, which can be toggled; see below. Full list of commands:

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

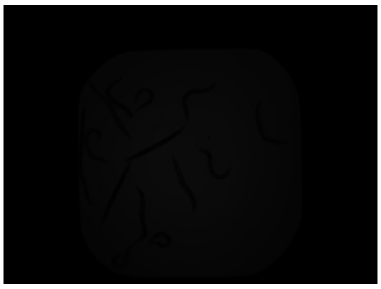
There are also boolean settings under the imports, which enable automatic saving of various images.
Processing of images is done in steps, where each step is a standalone method, all of which are documented and commented. The processing steps are as follows:

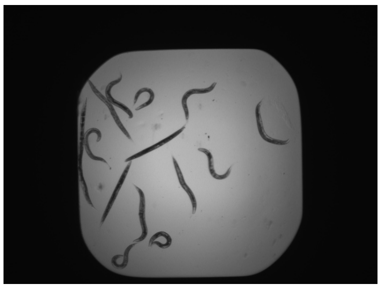
1. Isolate and compare
   (a) Isolate the worms from the background and border.
   (b) Compare the isolated worms against the provided ground truth.
2. Individualise and save each worm
   (a) Run the watershed algorithm to mark worms with unique colours.
   (b) Save each individual worm, under img_out/separated/ directory. This sub-step also prints the amount of worms counted.
3. Label each worm either dead or alive
   (a) Find the rotated bounding box of each contour, and classify dead or alive depending on the width/height ratio of this box.

Examples of what the above steps produce can be found under later sections. All images shown are of A01 w2, to clearly demonstrate the changes.

Choices Made

As the images are saved as 16 bit but only actually using 12 bits, when reading as 8 bit images the 4 most significant bits were lost, and the image was very dark (Figure a), 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 more usable, brighter image (Figure b).


(a) Loaded as 8 bit


(b) Loaded as 16 bit and shifted

I then adjusted brightness, and I found OpenCV's adaptive threshold to be much more effective than it's histogram equalisation, and the option to convert to a binary image proved useful.

The next choice I made was border removal for w2. My initial attempt found all contours and filled the largest in black, assuming this was the border. This worked for most images, although big worm clusters occassionally had more area than the border, and so were removed instead. I instead settled on dilating a flood fill that started from the point (0, 0). This meant that it would surround the border, and slowly grow over it. Experimenting with the values given to this function resulted in excellent border removal.

When comparing my binary images to the provided binary images, I decided to simply subtract one from the other, so that the resulting image's white pixels were representative of the error of my conversion. While this always results in high percentage values due to large empty spaces, and a big border surrounding the plate itself, other methods seemed to always provide no measure of how big or small their result actually was, just numbers without context.

To detect which worms were dead, and which were alive, I initially tried using contours, which could be checked by a method called isContourConvex, as this would indiciate whether the worm was curved or not. However, this method was inconsistent, and unreliable in actually providing a boolean return, so I changed to finding the bounding box of the contour, and checking whether it is long and thin (dead), or closer to square (alive).

Evidence of the success of the system in performing the specified tasks

Processing Steps

The loaded image, after shifting but before any of the major processing steps, is shown below as Figure 1.

Step 1, which isolated the worms from the background and border, produced Figure 2.

Step 2, which isolated individual worms, produced Figure 3 where worms are all outlined in red, and Figure 4 which has each worm coloured a different shade of grey.

Step 3, which drew boxes around worms, and coloured them green if considered alive, red if considered dead, produced Figure 5, the final image which is shown on screen during runtime.

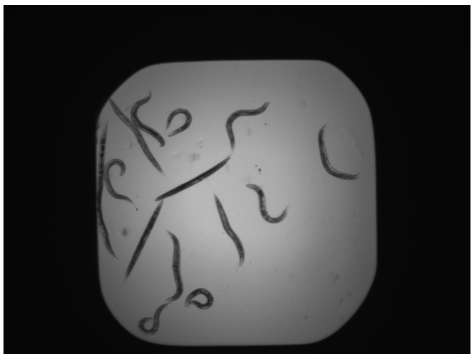


Figure 1

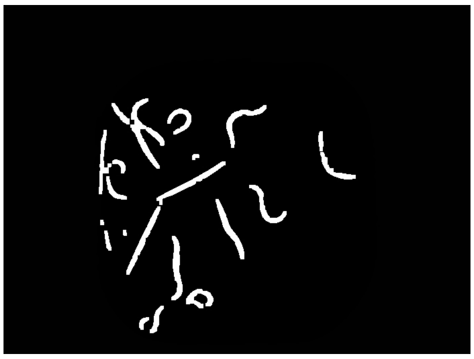


Figure 2

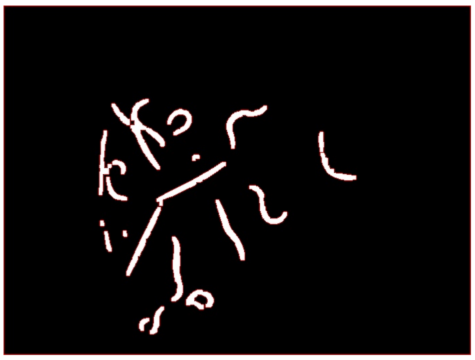


Figure 3

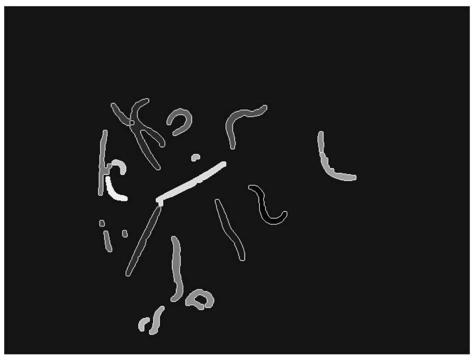


Figure 4

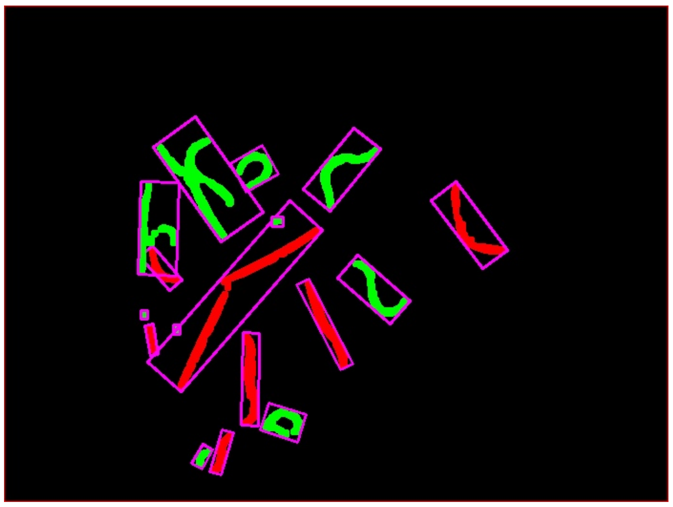


Figure 5

Further results images, along with individual worms images, may be attained by setting the relevant boolean settings to True at the top of the script.

Fault Cases

When a collection of worms overlap, the overlap is identified as one worm. Attempts made to identify separate worms that are touching were unsuccessful. If two or more worms are clustered closely but not actually touching, the script often manages to separate them, although this does depend on the size of the gaps between them. Finally, when thresholding, sometimes one curved worm is split into two, which will increase the worm count, and the two parts could be classified differently during dead/alive classification.