Bekzhan Kaspakov
Nurzhan Sakenov
Madiyar Katranov

Evaluation Document - Biological Image Segmentation

Biological image segmentation plugin for ImageJ is an image segmentation tool built using image processing algorithms, SciJava ecosystem of the FIJI/ImageJ2, ImgLib2 and SCIFIO libraries.

Testing/Evaluation

In order to test existing functionality we had to examine each part of the project for it to work properly. Most of the functionality consists of image processing algorithms, therefore in order to test the project we had to test the validity of each of the following algorithms. Simple unit tests were used to check the correctness of outputs:

Adaptive Denoiser:

The Adaptive Denoiser is a wrapper for ImageJ's own AD function. We checked that the code correctly calls IJ's methods and produces the expected output. However, due to the function's large complexity, we decided not to use AD in the standard pipeline.

Average Filter:

This algorithm is also used to denoise images. It works by assigning each pixel entry a new value of an average value of pixels surrounding it. This part of the algorithm was tested thoroughly by checking whether a certain input produces the desired output.

Binarization:

This part of the process does the job of creating binary images. The correctness of this algorithm heavily depends on finding out the correct threshold for distinguishing whether a pixel is going to be white or black. Therefore it required adaptive thresholding for binarization of different images. The correctness of this algorithm using adaptive thresholding was tested by assessing output images. The procedure did not produce plugin-specific bugs, except for a rare unexpected inversion of output that seems to be a bug internal to ImageJ itself.

Gaussian Difference:

This algorithm is used for increasing the visibility of the edges on an image. It does so by subtracting one blurry version of an original image from another lesser blurry version of the image. It was tested in the same way as the previous algorithm by assessing output images that come from running this algorithm.

Hybrid Filtering:

This is another method of denoising input images. This was tested and evaluated to work correctly by checking whether input pixel values produce proper output pixel values.

Local Entropy Filtering:

The entropy filter can detect subtle variations in the local gray level distribution. Therefore it is used to enhance our methods of edge detection. It was assessed to work correctly.

Median Filtering:

This algorithm is also used to denoise an image. It works by assigning each pixel entry a new value of a mean value of pixels surrounding it. This part of the algorithm was tested thoroughly by checking whether certain inputs returned desired outputs.

Normalization:

This part of the algorithm is used to control the range of pixel intensity of images that undergo processing. This algorithm was tested thoroughly by checking whether certain inputs return desired outputs.

Image Squaring:

The algorithm here is used to assign each pixel value the square its original value. It works correctly and requires minimal testing.

Tophat:

In our programming this algorithm is used for feature extraction and image enhancement. It does its job correctly as examined by various correct output images that it returns.

Apart from the algorithm itself, we also tested the usability of the base plugin (processing datasets using the standard pipeline). However, due to the pandemic, we were unable to perform the survey with biologists, as well as the major test on lab computers with datasets. Therefore, we were limited to manually running the program on datasets we had. We did not find any problems with the plugin itself. The images produced looked very similar to the segmented images provided by the Biology Department, but we did not have enough time to set up a comprehensive accuracy test.

Finally, the Graph GUI was tested using simple unit tests checking the correctness of graph construction and conversion from JGraphX to JGraphT (for easier graph traversal). As the module has not been finished to run the algorithms, we could not test the pipelines themselves, but they are constructed properly.