Self-Assessment – Reflective Piece 1
Computer Vision: Bio Image Processing

I have a keen interest in the processing of biological images such as cells. I think the area of computer vision will continue to grow and its application in health care could be a key component of improving health care. The image processing of cells can help measure and track cell attributes like mass, size, composition, movement, further analysis on goal post markers pre-applied to cell images etc. There are many variations and usage opportunities for these tools.

Depending on the logic and ML pipeline, these endeavours can help support biological research which in turn can lead to the early diagnosis of disease where symptoms and standard tests (urine, blood, and faecal matter etc.) do not lead health care practitioners traverse the issues some patients face. Patients not be aware of their predisposition to certain risks which can only be spotted a cellular level.

I have previous experience in image processing with languages/tools such as Java and MATLAB. I wanted to see what Python has to offer in this domain. So far, after taking a related course on DataCamp - I found the 'skimage' package in Python to be very useful with many useful functions and libraries that reduce the amount of code required to do some powerful image processing operations.

I would like to provide an example of something I processed and plotted today in Python. A few hours and very little code later, I found the result I wanted in Python with an equivalent body of work in MATLAB to compare against.

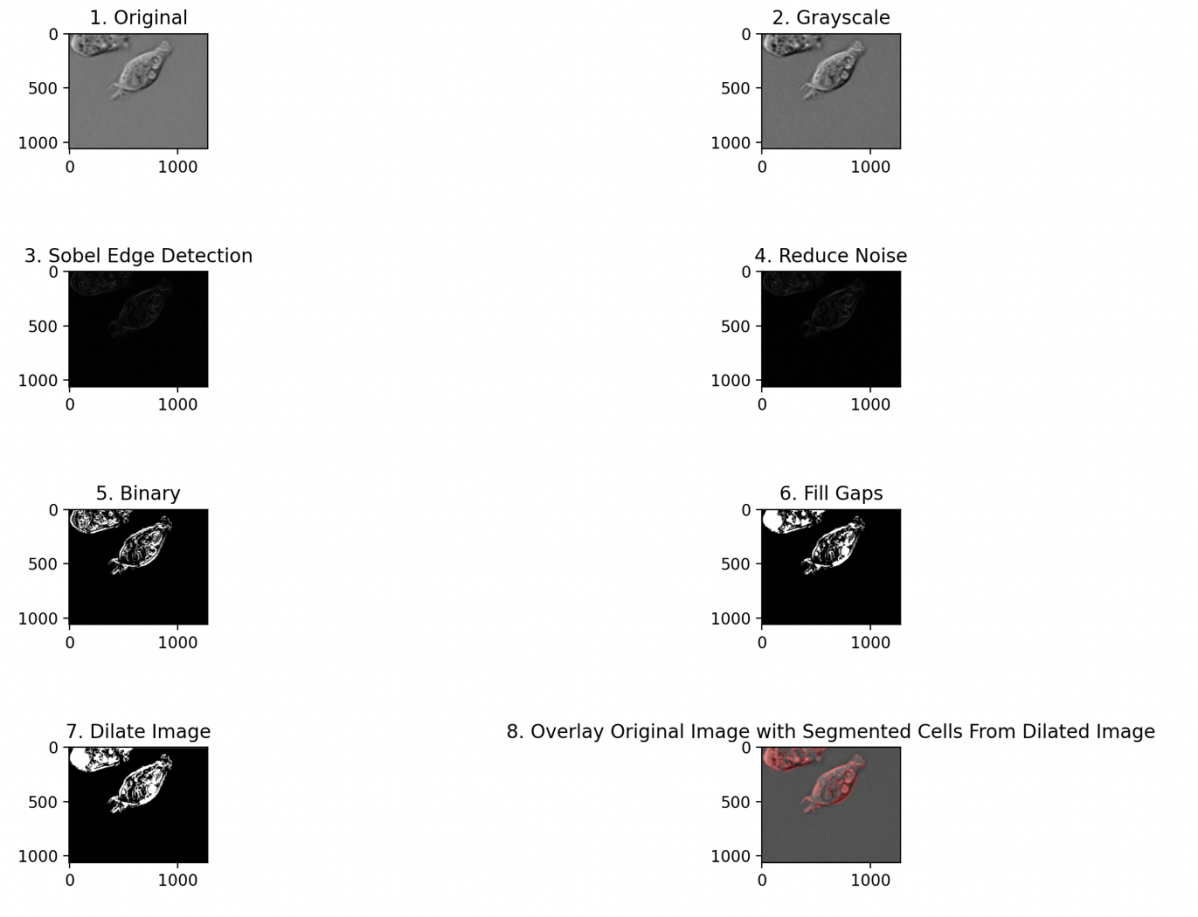
Please see the image produced from my script below which is a quick and dirty example of cell segmentation, the result is an overlay image to help convey how accurate this labelling process is with very little code and minimal time assigned to it.

The most time consuming steps were to select an appropriate thresholding algorithm followed by a suitable level of noise reduction from the image.

See final code snippet below:

```
thresh = ski.filters.threshold_triangle(sobel_edge)

red_noise = ski.restoration.denoise_tv_chambolle(sobel_edge,
            weight=0.01, channel_axis=-1)
```



The flow of logic was to:

1. Read the original image
2. Convert the image to grayscale
3. Apply Sobel Edge Detection (roberts, scharr and prewitt made little to no difference)
4. Reduce Noise (white spots outside cells)
5. Amplify image 4 by converting to Binary Image (*Note*: threshold_otsu() was not great option here)
6. Fill Gaps in Cells
7. Dilate Image to further Amplify
8. Create resulting image where figure 7 is the red overlay on original cells image

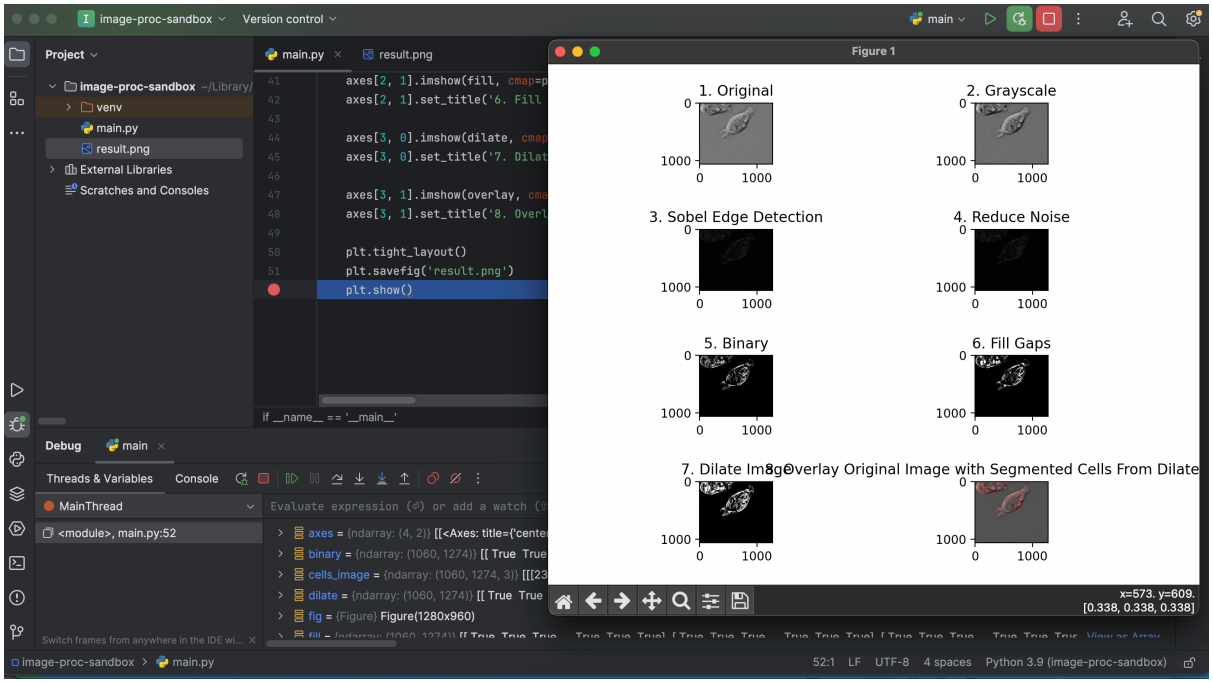
My aim was to visualise the segmentation of the cells image.

Reference:

1994-2023 The MathWorks, Inc. “Detect Cell Using Edge Detection and Morphology ”. Accessed: November 29, 2023. Available: https://www.mathworks.com/help/images/detecting-a-cell-using-image-segmentation.html

*View from **PyCharm** IDE:*

- *Note: The resulting image is saved as ‘result.png’ in working directory.*
- *The figure displayed can be resized / edited as needed and exported.*



*View from **VS Code** IDE:*

- *Note: Depending on python 3 installation, you may need to run package installer (pip) to install `skimage` package:*
  - *python3.11 -m pip install --upgrade pip*
  - *python3 -m pip install scikit-image*