Cell Colony Imaging - chapter 1

Felix Adams

March 19, 2024

1 Aims

One crucial aspect of quality control for induced pluripotent stem cells (iPSCs) is determining the confluence level. Confluence refers to the percentage of the area within a plate or well that is occupied by cells. In the iPSC pipeline, accurate confluence prediction is vital for estimating cell colony counts, which are essential for subsequent steps in the pipeline. The objective of this project is to develop a pipeline capable of accurately predicting cell colony counts based on bright-field images. Specifically, we aim to predict the confluence of cell colonies on a 6-well plate.

2 Introduction

There are a range of methods to predict cell counts in a colony. firstly, and most rudimentary, is manually estimating from eye. This is flawed on many fronts, but mainly due to user bias and a simple lack of compute.

Therefore computer vision methods have increasingly been implemented to carry out such tasks. One of the most popular methods is through deep learning methods and use of neural networks to classify colonies form non colonies. The advantage of such methods is the consideration of rules and deep complex relationships that are likely missed in other methods. But this also makes it a very difficult to understand the nature of the decision making, thus becoming a black box. Additionally, and perhaps most relevant to Cellected, is the extremely large quantities of labelled data to make such a method feasible. This is extremely time consuming and potentially expensive to perform. Therefore we will be using a combination of already made machine learning methods and basic computer vision methods.

3 Methods

3.1 Pixel classification

We use the pixel classification software from Ilastik. This works, in short, by implementing user defined colored labels on areas of a bright field image. for example, areas of cell colony are labeled in blue, the non colony area is labelled yellow, and the outline is labelled red. This fine tunes/trains the model so that when it sees similar pixels it will classify it as one of the predefined colors. We fine tune the model and can use it in the command line. This means we can run this model on many images in one go.

The output from this is an image with labelled pixels, where a pixel is either green, red or blue. Whilst this is promising to start, it is not the finished product. This can be best described as noisy and fuzzy and does not clearly distinguish colonies from the background.

3.2 Gaussian blurring

Gaussian blurring is a valuable preprocessing step in image analysis pipelines for several reasons. Firstly, it effectively reduces high-frequency noise caused by various factors, in this case the occasional misclassification of certain pixels. Secondly, it smoothes intensity transitions, facilitating more accurate edge detection by algorithms like Canny edge detection. This ensures that prominent features are highlighted while removing small details that may cause noise. Additionally, blurring enhances

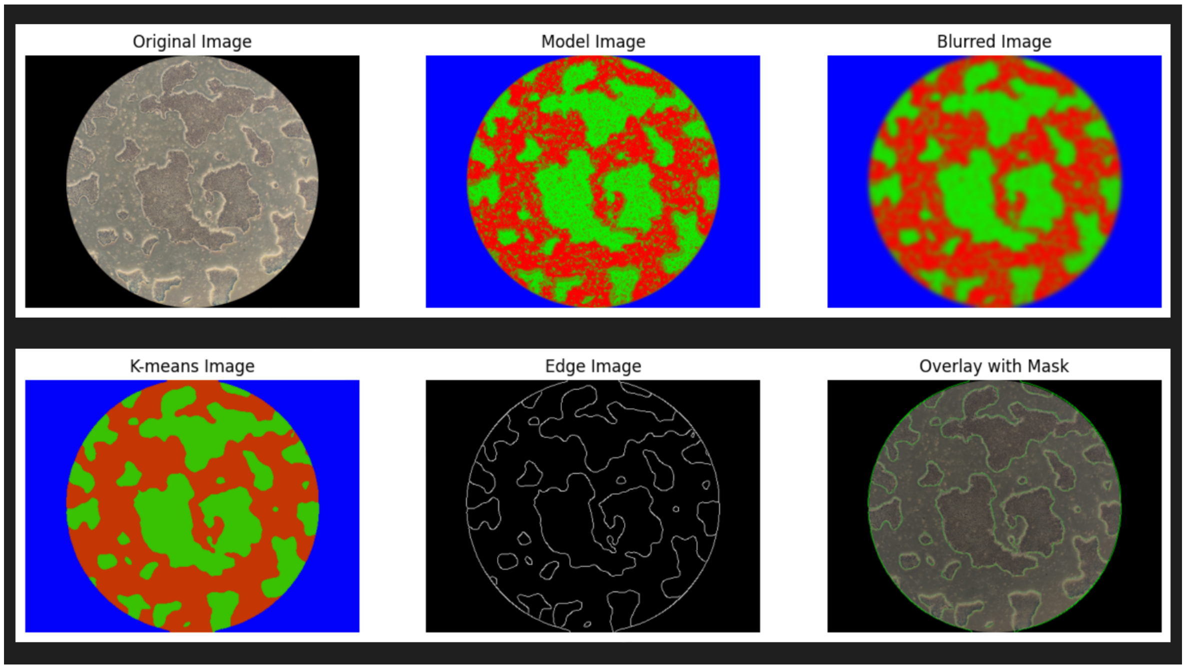

Figure 1: The various stages of image processing: (a) Original Image, (b) Model Image, (c) Blurred Image, (d) K-means Image, (e) Edge Image, (f) Overlay with Mask. These images depict the stages of processing applied to the input image.

subsequent processing steps such as segmentation or feature extraction by simplifying the image while preserving essential structural information. This optimization often leads to improved performance and accuracy in downstream tasks.

3.3 K-means Segmentation

We utilize K-means segmentation to group pixels based on color similarity, effectively assigning each pixel to a specific color cluster. This process helps in noise reduction by eliminating isolated pixels and preserving larger groups of pixels. By analyzing the surrounding colors, K-means segmentation aids in distinguishing between foreground and background, which is particularly useful for edge detection and assessing the proportion of colors in the image.

3.4 Confluence Calculation

The confluence of a given photo is computed by determining the percentage coverage of green pixels relative to the total number of green and red pixels. In other words, it measures the ratio of cell colony area to the combined area of cell colonies and non-cell colonies.

3.5 Canny edge detection

In our image processing pipeline, we leverage the Canny edge detection algorithm to detect edges within our images, a critical step for delineating object boundaries. This method begins by converting the input image to grayscale to simplify the analysis, followed by gradient calculation to identify areas of significant intensity changes indicative of edges. Subsequent non-maximum suppression ensures that only local gradient maxima are retained, resulting in thin, accurate edge representations.

3.6 Superimposition of edges to original images

we superimpose our predicted edges to the original image. This allows to assess the efficacy of the model.

3.7 Conclusion

We have the first version of a model capable of calculating the area of the cell colony with a bright field image.

There is plenty of opportunity for tailoring this pipeline and room for improvement.