Counting Bacteria in a Culture Using OpenCV

Alexander Cardaras

I. Introduction

Given an animated GIF file of bacteria being tracked by a growth and motion algorithm, I was able to implement a solution to automatically estimate the number of bacteria present in a given frame. The solution I propose accomplishes this task by tracking the movement of each bacteria as it progresses through its life cycle. A hierarchical representation of the bacteria is constructed by linking daughter cells to their parent cells after cellular division has occurred. Because of this, it is possible to predict the movement of each bacteria in each subsequent frame. The advantage of this is that it allows for the detection bacteria that the initial tracking algorithm may have missed.

II. Assumptions

Before beginning this challenge, I made an assumption on what a fully split bacteria looked like. I assumed that bacteria do not have to appear to be completely separated in order to count as two individual bacteria. For example, I make the assumption that the rightmost entity in *Fig. 1* is actually two separate bacteria overlapping.

III. OpenCV Image Manipulation

While the growth and motion algorithm provides a strong foundation to start image analysis, there is still room for improvement. The major shortcomings I observed while looking at the images were as follows: The image contains noise that could potentially be marked as bacteria. Some bacteria's outlines have gaps or holes in them. There are many cases where bacteria are outlined as a single entity but in reality are two, close together, bacterium.

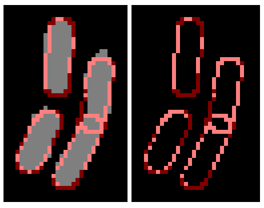
Fig. 1

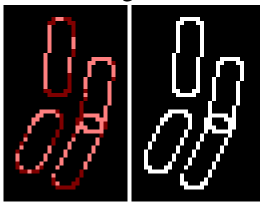
Fig. 2

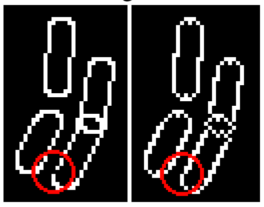
Fig. 3

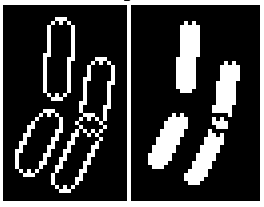
Fig. 4

Steps for Preparing the Image

Image Masking (Fig. 1): The easiest way of removing noise from each frame is to mask out all of the non-red pixels. In the image, red pixels outline bacteria that the growth and motion algorithm found. By simply changing the color of all non-red pixels of the image to black, we end up with red outlines of bacteria on a black canvas.

Binarization(Fig. 2): Convert the image to gray scale and replace all non black pixels with white color values to prepare for further operations.

Dilation and Skeletonization(Fig. 3): By dilating the image with a small kernel size of 3x3 over a single iteration, the bacteria outlines thicken and small holes are filled. Following this with skeletonization reduces the outlines of the bacteria back down to single pixel wide lines. What results is a thin outline of the bacteria similar to the binarized image, but with holes filled in. The skeletonization algorithm I used erodes the image with a 3x3 morphological cross structuring element until the outline's thickness is reduced to one pixel. Note that using non-cross shaped structuring elements may not fill holes.

Bacteria Isolation(Fig. 4): The final morphological operation fills the bodies of the outlined bacteria and discards the original outline. This is achieved by performing a bitwise not operation on the image followed by flood filling the background of the image. The advantage of working with filled bodies over outlines is that the filled bodies do not intersect, making it simpler to differentiate between one large bacteria and two smaller overlapping bacteria.

Locating Bacteria

Find Bacteria Contours: After all image filtering is complete, getting a general count for the number of bacteria in the frame is as easy as using the find contours function and filtering by contour area. Notice, in *Fig. 4*, the creation of a small white body at the intersection of the two rightmost bacteria. Although this body is not a bacteria, the find contours function will still return it as a possible bacteria candidate. Filter the contours by area and discard any contour whose area is relatively small.

IV. Bacteria Tracking

Bacteria Position Predictions

Before the current frame is analyzed, predictions on the locations of the previous frame's bacteria are made. This is accomplished by taking the average velocity of each bacteria and applying it to their last observed/estimated location. In special cases where bacteria have no past location data to compute an average velocity, due to cellular division occurring, the two daughter cells adopt their parents velocity and an additional force applied. This additional force is generated from the division process itself, propelling both daughter cells in opposite directions from each other. An exaggerated illustration of these forces can be seen by observing the the cyan colored bacteria in *Fig. 5*. In this example, colored lines represent bacteria's bodies and the pink dots represent their centers.

Once the locations of all bacteria from the previous frame have been predicted, they are compared to the positions of bacteria in the current frame. Position data of bacteria from the previous frame is overwritten by the nearest bacteria in the current frame. Essentially snapping the locations of each bacteria from the previous frame to the locations of bacteria in the current frame. Each bacteria from the current frame can only be paired with a single bacteria from the previous frame. Sometimes this results in a previous frame's bacteria not being paired with a bacteria in the current frame. In this scenario, the bacteria from the previous frame keeps it's estimated position.

Cellular Division Estimations

Fig. 6 shows one of the many times that the growth and motion algorithm counted two bacteria as a single bacterium. The inconsistencies with this algorithm make it difficult to determine when a bacteria has undergone a cellular division. As a result, the growth and motion algorithm is used as a secondary indicator to determine if a bacterium has divided, with the primary indicator being the length of the bacterium. Once a bacterium's length exceeded its width by sevenfold, it is considered to have split even if no physical gap between the two bacterium exists. Once this split occurs, the daughter cells are treated as two separate entities with unique positions and velocities, even if the growth and motion algorithm does not reflect this.

V. Conclusion

This solution to the automated bacteria counting problem builds off of a growth and motion algorithm. By performing morphological operations with the OpenCV framework, I was able to filter out noise in the image and fill in the gaps in the outlines of the bacteria. Bacteria position's were found using OpenCV's find contours function. Estimating the positions of bacteria proved to be useful for tracking the location of bacteria as they grow and move around in each frame. Location tracking was used to make up for some inconsistencies present in the growth and motion algorithm regarding cellular division.

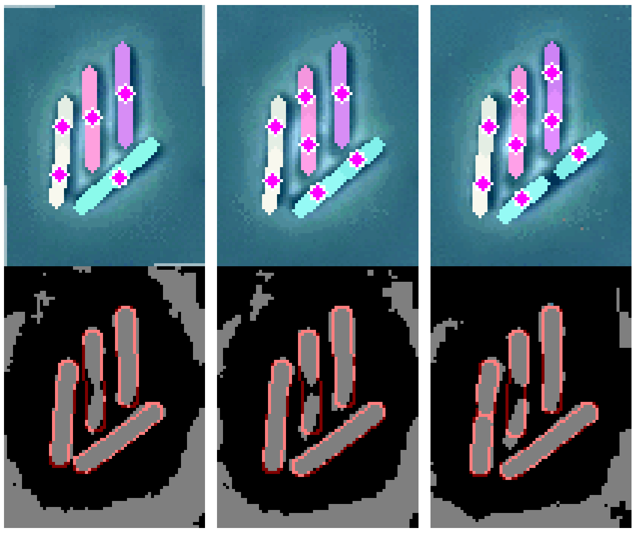
Fig. 5

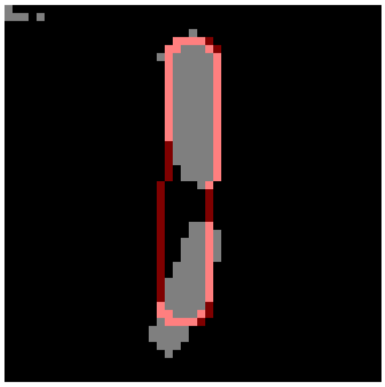
Fig. 6