A System for Automatic Cell Segmentation of Bacterial Microscopy Images

Eric Battenberg and Ilka Bischofs-Pfeifer

***Abstract*—We explain a system for automatic cell segmentation that uses Matlab and the free DIPImage package...**

***Index Terms*—image processing, computer vision, segmentation, bacteria**

I. Introduction

In order to process the large number of bacterial microscopy images produced by the Subtilis group at the Arkin Lab, we have developed an automatic cell segmentation system with the help of Matlab and the DIPImage toolbox. As with any segmentation system, we desire that the area containing each individual cell be clearly identified. In addition, error rate, execution speed, and amount of user intervention are important factors.

Our approach combines many existing techniques that are suitable for our specific application. The signal-to-noise ratio of typical test images, shape of B. Subtilis cells, and the structural characteristics of the microscopy images are exploited to make the system more effective.

The system uses a three step approach. First, clusters of cells are segmented from the background. Then the individual cells are segmented within each cluster. Last, segments containing multiple cells are split based on the shape of the segment. Prior to the segmentation steps, a preprocessing routine is used to remove noise and fix contrast and lighting.

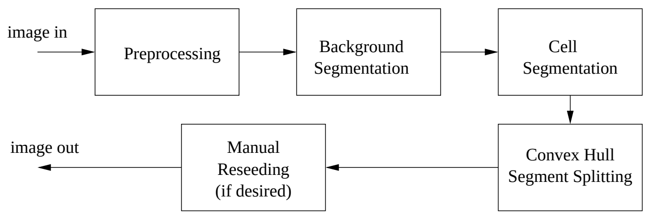

Fig. 1. The System

II. Motivation

The motivation behind the development of this system comes from the lab's need for automatic processing of the many microscopy images produced in experiments with the bacteria B. Subtilis. Also, a lack of a freely distributed bacterial segmentation system is of equal concern. The system is written as a Matlab class with the help of the image processing toolbox DIPImage (free for non-commercial use). The use of DIPImage makes the purchasing of expensive Matlab toolboxes unnecessary.

In our lab, the system will be used to acquire a clean segmentation from phase contrast or bright field microscopy images and then use the segmentation regions as a mask for measuring the intensity of different fluorescent proteins. Eventually, we hope to augment the system to be used in video lineage tracking as well.

Eric Battenberg, Graduate Student, Dept. of EECS, University of California, Berkeley
Ilka Bischofs-Pfeifer, Post-Doctoral Researcher, Lawrence Berkeley Lab

III. Preprocessing

The proprocessing step uses some interesting techniques that are tailored to improve the effectiveness of the specific type of segmentation methods used in the later steps. After the input image is read into memory, its intensity values are linearly stretched between 0 and 255. Any floating point precision present in the input is retained. The stretching is done so that common parameters can be used for all images of all intensity ranges. If necessary, the image is inverted in order to make the cells of higher intensity than the background. Then a power-law transformation is applied to increase contrast in the bright cell regions.

Following the contrast operations, homomorphic processing is used to compensate for uneveness in lighting [1]. This technique relies on the luminance times reflectance model of image formation. To filter out the luminance portion of the product, we take the log of the image to allow for a simple linear separation of the two components. By subtracting a version of the image strongly blurred with a Gaussian kernel, we can eliminate much of the slowly varying luminance component of the image. The process is completed by taking the exponentiation of the resulting image.

Denoising is accomplished using the Vincent Dome Transform [2], or h-dome, as suggested by Kutalik [3]. This transform is a morphological process that relies on the greyscale reconstruction operator. If the input image is viewed as a topographical surface, the transform removes domes up to a certain specified height. The removed domes then leave behind a flat surface. Since the allowed removal height is chosen to be much less than the difference between cell intensity and background intensity, the transform preserves the structural edges of the cells while removing smaller variations caused by noise and internal cell structure. The preprocessing is completed after a 3x3 median filter (a nonlinear filter effective in the removal of salt and pepper noise) is used to clean up any large noise spikes that were not removed by the previous transform.

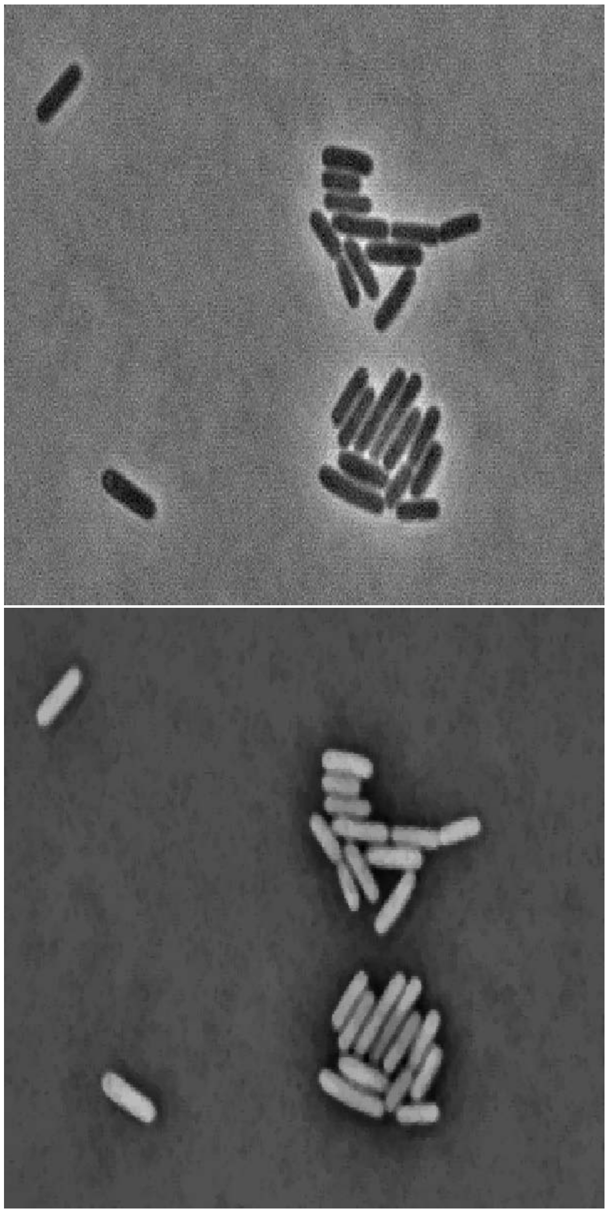

Fig. 2. Input image and preprocessed image

IV. Background Segmentation

Background segmentation is done using edge detection and region growing. Because a sharp increase in intensity exists around the edges of a dark cluster of cells in phase contrast images, the top hat morphological operator is used to detect cluster edges [1]. The top hat transform is used for enhancing detail and detecting convex objects smaller than a certain size. The threshold for the edge detector is automatically chosen using the Otsu algorithm [4] which uses the image histogram to minimize the variance above and below the threshold.

After the edges are detected, the skeleton morphological operator is used to shrink the think edges down to thin border lines. These border lines now define the separation between the cell clusters and the background. To separate the background from the clusters, we first define background seeds that are grown but constrained by the border lines. The background seeds are chosen to be areas that are less than some background threshold. This threshold is chosen automatically to be the mean of the image, but can be manually chosen as well. The resulting seed areas are then morhologically eroded a few times to reduce accidental overlap with cell clusters.

A fill routine that employs binary propagation is used to grow the seeds until they hit the cluster borders. Inverting this filled region results in a binary image containing the cell clusters.

One problem with this approach is that filling in this manner includes small internal gaps in the cell clusters. Our solution is to use a smaller threshold to locate the extremely dark internal gaps within the clusters. These gaps are then eliminated from the cluster image.

Lastly, clusters smaller than a certain size are removed from the binary image.

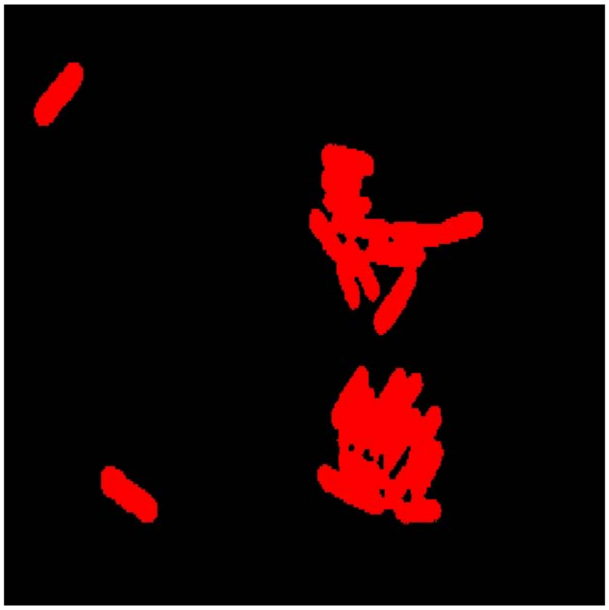

Fig. 3. Clusters segmented from background

V. Cell Segmentation

Our next task is to segment invidividual cells within each cluster. Again, the basic idea behind this task is seed selection and region growing. A separate seed is define for each cell, and the seeds are expanded using the seeded watershed algorithm [5]. The seeded watershed algorithm works very well given that the seed are chosen correctly, so seed selection becomes our primary concern.

To define the seeds, we again use the top hat transform to detect edges. The edge threshold is the important parameter here. By default, we have chosen to use the Otsu threshold of the top hat image. While this threshold provides acceptable results for a completely automatic segmentation, it frequently causes an undersegmentation of the cells. Because of this, a manually chosen edge threshold for each set of images with similar statistics is desired.

By decreasing the threshold, segmentation tolerance is decreased – meaning more segments will be found – and increasing the threshold causes less segments to be found.

After the edges between cells are detected, the fill routine is used to fill the inside of the detected edges. This routine also eliminates the edges themselves from the resulting binary image, so we hopefully end up with a bunch of separate seeds located at the center of each cell. If the edge threshold is chosen to be too low, details within each cell will be detected as edges which could result in multiple seeds being found within a single cell. If the threshold is chosen to be too high or a weak separation exits between cells, a single large seed may end up spanning two or more cells. In either case, manually adjusting the edge threshold until a good segmentation is attained is all that is required. A threshold arrived at in this manner will usually suffice for all similar images captured under the same conditions.

Following seed selection, we use the seed watershed algorithm. The watershed algorithm [5] is an important technique used in image segmentation. To understand the algorithm, we first must picture a 2D image as a topological intensity surface with bright regions at higher altitude. Within this surface, we can visualize water being pumped into each local minima. As the water rises, the water coming from two separate local minima may eventually merge into one body of water. To prevent this we build a levee, or watershed, between the two bodies of water. This filling process is continued until all possible watershed lines are drawn in the image. These lines can then be used as segmentation boundaries. The problem with this, however, is the obvious oversegmentation that arises when lines are drawn between all adjacent local minima.

The solution to this is the seeded watershed algorithm. This process differs from the original algorithm in that water is only pumped in at each seed location rather than at every local minima. Watershed lines are drawn to prevent the merging of the bodies of water belonging to two separate seeds. In this way, cell boundary lines can be drawn and each seed can be grown out to its boundaries.

The actual surface image we use in this seeded watershed process is our top hat transformed image which possesses stronger edges and less brightness variation than our original image. The output of this process is an image comprised of integer values which label the region containing each separate segment.

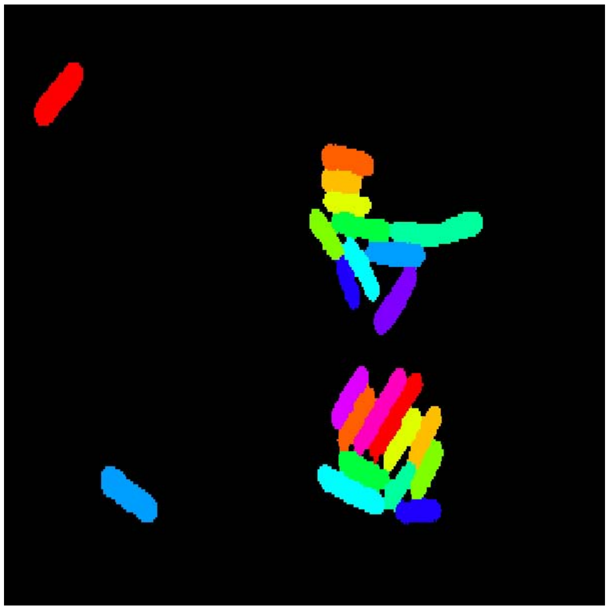
Fig. 4. Segmented cells

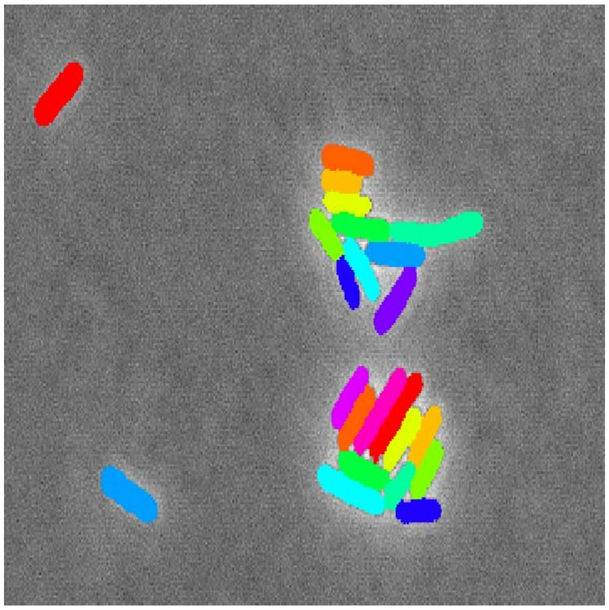
Fig. 5. Segments overlaying input image

VI. Postprocessing

In the postprocessing stage, we wish to correct under and oversegmentation. In both cases we attempt to correct the initial seeding and then regrow the segments from the new seeds. Automatic correct is the first step. This step uses a convex hull segment spitting algorithm devised by Kutalik [3]. Following automatic correction, an optional manual reseeding step can be taken.

A. Automatic

To automatically divide undersegmented cells, we use Kutalik's convex image segmentation algorithm [3]. This algorithm uses the shape and convexity measure of a segment to draw a rupture path through the undersegmented region. It is particularly useful considering the shaping of bacterial cells and the typical arrangement of adjacent cells.

The algorithm first calculates a transformed image for the segment in question which is the distance between the border and the convex hull of the segment at each point. This new image is used to calculate a rupture path which divides the segment in two. The algorithm relies on the fact that the intended rupture path is made up of points that are each a local maximum in the direction perpendicular to the path.

Once the path is decided upon, the original segment can be divided in two. For our purposes, though, we apply this algorithm to the seeds rather than the actual segments. The convexity of segments in our images is not usually great enough to ensure an accurate division. We also found that bridges between merged seeds are usually very thin making divisions more accurate.

To run this algorithm on every seed would take a very large amount of time. To narrow down the field, we use requirements on the Podczeck shape descriptors of each segment [6]. Once suspect segments are identified, the seeds corresponding to them are handed to the convex hull splitting algorithm. The newly separated seeds are used (along with the unchanged seeds) to regrow the regions using the watershed algorithm.

B. Manual

To correct any glaring errors in the automatic segmentation, we have included an option to manually reseed segments. Using the Podczeck shape descriptors again, suspect segments are shown to the user who can then locate each seed using the mouse.

After all the segments in question are presented, the user can click on any segment to zoom in and then correct any seeds in the immediate area. We do not expect this manual reseeding operation to be used much – especially in video applications where a highly automated process is desirable. However, when the highest level of accuracy is required, the user may have to take this step.

VII. Future Additions

While this system is fairly complete for simple bacterial cell segmentation, it must be augmented to actually gather scientific data from each segment. In the near future, we hope to implement interimage alignment and fluorescent channel measurements to be used in detecting the expression of certain proteins within each cell. We also hope to develop a lineage tracking system that utilizes the segmentation capabilities of this system.

VIII. Conclusion

The bacterial segmentation system presented here is tailored to the specific nature of the images and cells used in our experiments. Bright field and phase contrast images are the primary image types used to produce a good segmentation mask.

Each of the Matlab functions that comprise the system contain their own documentation. A sample script that uses the functions as intended is also available.

References

[1] R. Gonzalez and R. Woods, *Digital image processing.* Addison-Wesley Reading, Mass, 1987.

[2] L. Vincent, “Morphological grayscale reconstruction in image analysis: applications and efficient algorithms,” *Image Processing, IEEE Transactions on*, vol. 2, no. 2, pp. 176–201, 1993.

[3] Z. Kutalik, M. Razaz, and J. Baranyi, “Occluding convex image segmentation for E.Coli microscopy images,” in *Proc. European Signal Processing Conference (EUSIPCO)*, 2004, pp. 937–940.

[4] N. Otsu, “A threshold selection method from gray level histograms,” *IEEE Trans. Systems, Man and Cybernetics*, vol. 9, pp. 62–66, Mar. 1979, minimize inter class variance.

[5] S. Beucher, “Segmentation tools in mathematical morphology,” *Proceedings of SPIE*, vol. 1350, p. 70, 1990.

[6] F. Podczeck, “A shape factor to assess the shape of particles using image analysis,” *Powder Technology*, vol. 93, no. 1, pp. 47–53, 1997.