CS279 Project: Detection and tracking of deformable red blood cells

December 4, 2015

Leonid Keselman
leonidk@stanford.edu

We implemented a full tracking pipeline of red blood cells, including automated detection with a Hough Circle Transform, and deformable tracking with our own Python implementation of Distance Regularized Level Set Evolution (yes it was written just for this class!) on top of basic image processing primitives provided by Python's scikit-image library (e.g. Gaussian Filters). We analyze the behavior of this tracking system and its shortcomings.

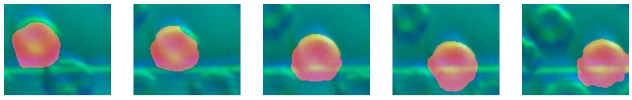

FIG. 1. An example of a tracked cell sequence. The original microscopy data is shown in a cyan colored background while the tracked segmentation result is overlaid in red. See results section for details.

INTRODUCTION

In generating and analyzing large amounts of data, automated methods are important tool in allowing researchers to obtain quantitative and statistically significant information. For microscopy applications, the data of interest is often detection and analysis of cells, including their geometry, shape and position.

For this project, we chose to focus on red blood cells, which are of interest to malaria researchers. The leading source of malaria infection worldwide is the parasite *Plasmodium falciparum,* which inflects red blood cells. The ability to automatically detect red blood cells would make it possible to automatically detect and check for the prevalence of the parasite in microscopy data.

Additionally, we focused on tracking red blood cells which were placed in a waveguide constructed by Saara Khan & Kara Brower, from the Fordyce & Solgaard Labs at Stanford. We are acknowledge and are thankful for their data. In the conditions of the waveguide, there is often squeezing of the red blood cells, which provides useful information about deformability, which can be an indicator of *P. falciparum* infection. However, this requires deformable tracking of the red blood cells, which is still an open problem in the image analytics literate **[6]**.

I. RELATED WORK

There have been several published methods of tracking red blood cells from that past few years.

One example from 2012 **[7]**, focuses solely on detection of red blood cells using a Hough Transform. In the course of this work, the Hough Transform is used as both an automated detection and post-processing method. See the section on detection.

Another example from 2014 **[8]**, focused on tracking red blood cells in very noisy environments, as their data is from intravital microscopy.

A general deformable tracking method using active meshes was documented in 2011 **[6]**. This allows for tracking of surface contour elements across the entire sequence. Since this application only requires identification and segmentation of results, we decided to focus our energies on a simpler method.

A classic area of object contour detection is active contour methods (also called snake methods). Geodesic active contours have been shown to be successful in demonstrating active tracking in the medical literature, specifically for tumor and organ segmentation **[9]**. A recent formulation of this approach, called Distance-Regularized Level Sets **[1]** demonstrated an implementation of active contour methods without the need to reinitialize. As it is always preferable to have one less parameter to tune, we decided to implement that method as our primary form of deformable tracking.

II. METHODS

A. DETECTION

I. HOUGH CIRCLES

Detection of the cells is done via a Hough Circle Detector. A Hough Detector works by first detecting edges, for which we use a canny filter. See Figure 2. Afterwards, it counts contributions for all positions and radii of interest and picks the most common hypothesis. The radii of interest are selected manually (assuming cell size is roughly constant). The Hough Circle Detector is from the Python scikit imaging library.

II. TRANSMITTANCE MICROSCOPY

As seen below in Figure 2, the Hough circle detector can successfully detect red blood cells, even with very bad edge detection from the canny edge detector. In the image below, from the Yeh Lab at Stanford (data from Katie Amberg-Johnson), we see the detector able to handle a very challenging case in normal transmittance microscopy. All four blood cells, despite their inner geometry are detected correctly and identified. Even the localization is very good,

III. REFLECTANCE MICROSCOPY

The data from the microfluidics experiments, however, is from a different from of microscopy, namely reflectance microscopy. In this experimental condition the illumination interacts with the geometry of the red blood cells to a much higher degree, creating many false edges. In this tracking sequence, only the top one cell is chosen to be tracked and detected. Additionally, multiple radii sizes are chosen and all of them are used to create an initial level set for the sequence (see next section).

FIG. 2. This is an example of red blood cell detection in transmittance microscopy, with a circular Hough detector and four red blood cells. The left frame is the original image, the middle frame is the canny detected edges, and the right frame are the top 4 detected Hough circles overlaid on top of a washed out version of the original image.

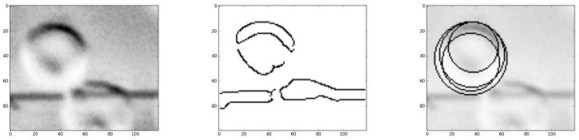

FIG. 3. This is an example of red blood cell detection in reflectance microscopy, with a circular Hough detector and two red blood cells. The left frame is the original image, the middle frame is the canny detected edges, and the right frame are the detected Hough circles overlaid on top of a washed out version of the original image.

.

B. TRACKING

For tracking, we only worked off the microfluidics waveguide dataset (as seen in Figures 1 and 3). This is a dataset which requires deformable tracking as the waveguide can sometimes squeeze the cell.

In the case of deformable tracking, we implemented the Distance Regularized Level Set Evolution paper **[1]**. Our implementation was done from scratch in Python but we mirrored at the original authors MATLAB implementation to catch errors and ensure correct implementation.

I. LEVEL SET METHODS

Level set methods are a class of contour detection where instead of tracking the edges directly, edges are encoded implicitly through a signed distance field. In a signed distance field, every pixel stores its distance from the closest edge. Negative values are “inside” and positive values are considered “outside”. Therefore, the value matching numerical zero is the position of the edge.

In active contour methods, the segmentation is seeded with an initial boundary region, from which it is iteratively evolved towards the local minima of edges. There are multiple parameters, including convergence step size, but the one found most important was alpha. Alpha controls how the level set evolves, and also which direction.

II. INITIALIZATION

Using the Hough circle as described in the detection section, the level set is seeded with the interior of the Hough circle as the inside of the cell. That is, all pixels inside all valid Hough circles are used. Then it is evolved to convergence.

III. LEVEL SET EVOLUTION

Level sets can either be evolved outwards (from a small seed region) or inwards (from an oversized seed region). To handle this explicit restriction, we always either dilate or erode the initialization to make sure the initialization is correct for which form of level set evolution we're using. When we're evolving outward, we shrink the initialization region; when we're evolving inward, we grow it.

We tried examples of both evolution methods and with parameter tuning were able to obtain similar results. However, we found that the level set method sometimes locks onto the edges of the image frame, so we preferred the outward growing method for most of our experiments.

The level set evolution can be seen as a gradient-descent search for closest edges, and in the low resolution, noisy images we used, there are many edges which can satisfy a local search, so different initializations can sometimes lead to different results, as seen in Figure 3.

In order to perform tracking, one can simply take the previous frames' level set, erode or dilate it, and use that as the seen for the next frame to perform level set evolution. This behavior, repeated iteratively, leads to the behavior shown in Figure 1.

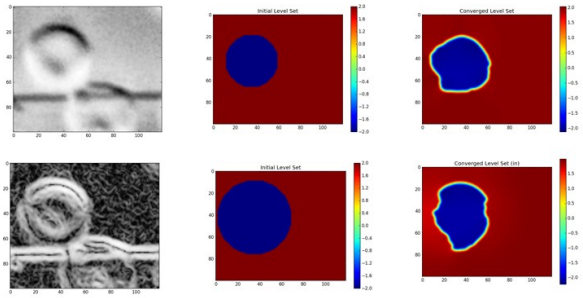

FIG. 4 This is an example of an image (top left) and its edge indicator function (where white are stronger edges) (log scale) (bottom left). The middle columns are the Hough circle initializers, eroded and dilated. The right column shows the final converged level sets for both conditions.

IV. POST-PROCESSING

There were two main issues that required post-processing in additional to the tracking method: false edges contributed by the waveguide and run-away errors due to image noise. We implemented methods to address both.

In order to handle interruption from the waveguide, we used a mean image over the image sequence (as the microscope is stationary) in order to provide an easy background image. However, this background image sometimes contains real edges from the image sequence, so we had to choose a blending factor for the mean image. We finally settled on 15%. This is shown in Figure 5.

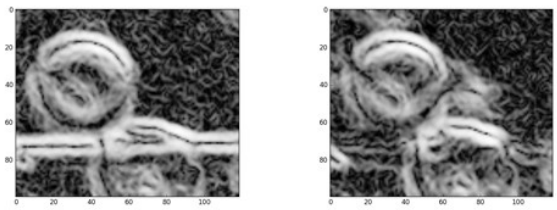

FIG. 5 An example showing the change of edge indicator functions when subtracting the mean image. The original is on the left and the subtracted image is on the right. As is clearly visible, the subtraction removed the waveguide but also creates false new edges and removes some of the real cell data as well.

Additionally, sometimes the edges run catch onto another cell or the waveguide and shown runaway behavior. To handle this, we run another Hough transform (either on the image data or the level set image, we found both worked) to detect the cell. We then use an oversized version of this Hough circle as a boundary limit, and remove all level set data outside the circle. This keeps the contours tracking even in the case of noisy edges. See Figure 6 below.

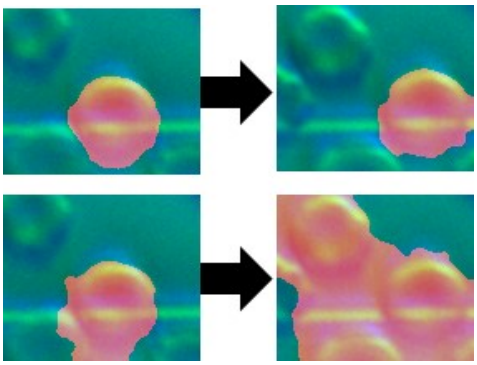

FIG. 6 Runaway behavior of level set tracking. Sometimes the contours' poor converge (bottom left) can lead to disastrous results later (bottom right). Using the Hough circle post-processing technique described, we're able to control the errors in the level set and have the contained behavior seen in the top row.

III. RESULTS

Basic tracking results can be seen in Figure 1, where five example frames from an 84 frame sequence are shown (#1, #23, #45, #58, #84). Successful tracking is seen in this example as the cell starts roughly correct, and then transitions across the waveguide correctly. This required a combination of background Hough Circle initialization, background subtraction (15% of mean image), and Hough Circle on Level Set to post-process growth. This was done with an alpha = -0.8 (which is an outward growing level set), and dilation with a disk of size 3 was used between time-steps. This was the most successful run on this dataset; a discussion of more challenging issues is brought up in section **IV A.**

A. TRACKING SIZE

As a way to test how robust the tracking is, we plotted the size of tracked cell (in pixels) across the test frame sequence. Notable errors occur when the cell begins to interact with the waveguide (#30-#45) and when it gets near the edge of the frame (#75-#82). However, both errors are still fairly small relative to the side of the cell.

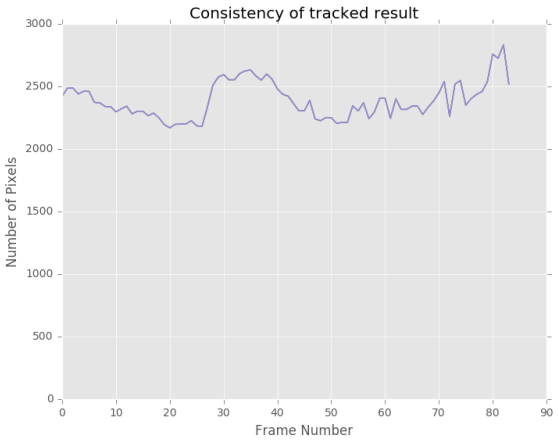

FIG. 7. A graph showing the consistency of a single tracked cell across an image sequence, as the cell crosses a waveguide at around frame 40.The sequence is that from Figure 1.

IV. DISCUSSION/CHALLENGES

A. PARAMETER TUNING

One of the biggest challenges in using this algorithm correctly is the enormous number of open parameters: how much of the mean image is subtracted, how high is alpha set, how large are the cells, how aggressive should the post-processing circles remove growing data and how much uncertainty is there in-between frames.

Some examples of poor parameter setting are shown in Figure 8. If there were additional time, it'd be nice to either find a more robust set of parameters or simply automate their setting. By exploring how this algorithm works on a larger data set, this should be possible.

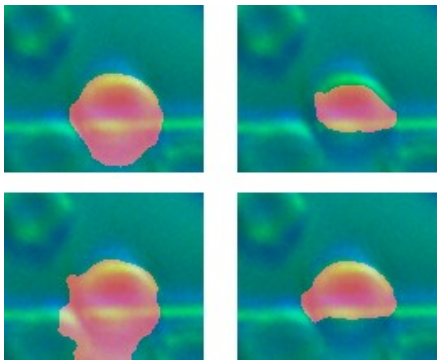

FIG. 8 Various answers given by the algorithm for the same exact frame (#58) depending on parameter tuning.

B. WAVEGUIDE

The waveguide is a large conflating factor in the edge indicator function, which drives all the segmentation. By exploring more robust methods of removing the waveguide (such as more sophisticated background subtraction techniques), we believe that our results would be better.

The challenging part of the waveguide behavior is that the waveguide often grabs the cell, and these are the most interesting scenarios. This is also the location of most expected deformation. Thus our handling of the waveguide must be very careful as to not remove the useful cell edges which are present in that area of the frame.

C. NON-UNFORM LIGHTING

Using the waveguide microfluidics data, we're in a reflective microscopy condition, and this makes simple contour tracking difficult. Compared to transmissive microscopy, where the edges are often illuminated and clear, reflective microscopy often has much murkier edges. As the edge indicator function is the basis of the segmentation method, this often causes erroneous segmentation and tracking.

There are two interesting methods we'd like to explore in order to handle this non-unform lighting. The first is to explicitly model and estimate the lighting, so we can create an edge indicator function that is more robust. For example, in these sequences, the top of the cell is always illuminated, and all edges there should be bright, while the bottom of the cell edge should be dark. If this information can be brought into the edge indicator function, we believe our tracking would significantly improve.

Another method we'd like to try would be to use a machine-learned edge indicator function. These are well documented **[2] [3]** in the computer vision literature, using Trees, SVMs, and CNNs. They're used in cases such as semantic segmentation, where image edges doesn't have a clear mapping to what people perceive as object edges. We

have a similar challenge. In our specific case, with some manual cell annotation, machine learned edge detectors might be used to handle both the non-uniform lighting and the waveguide with one fell swoop.

D. OTHER TRACKING METHODS

In addition to the deformable tracking, it'd be interesting to explore more long-term optical flow methods, which keep a history of cell appearance and can handle distortions that span multiple frames and are sometimes occluded.

Additionally, optical flow methods operate on image differences and might behave better in such low quality images.

---

1. **[1]** Li, C., Xu, C., Gui, C., & Fox, M. D. (2010). Distance regularized level set evolution and its application to image segmentation. *Image Processing, IEEE Transactions on*, *19*(12), 3243-3254.
2. **[2]** Bertasius, G., Shi, J., & Torresani, L. (2014). DeepEdge: A Multi-Scale Bifurcated Deep Network for Top-Down Contour Detection. *arXiv preprint arXiv:1412.1123*.
3. **[3]** Dollar, P., Tu, Z., & Belongie, S. (2006). Supervised learning of edges and object boundaries. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 2, pp. 1964-1971). IEEE.
4. **[4]** https://en.wikipedia.org/wiki/Circle_Hough_Transform
5. [**5**] Illingworth, J., & Kittler, J. (1988). A survey of the Hough transform. *Computer vision, graphics, and image processing*, *44*(1), 87-116.
6. **[6]** Dufour, A., Thibeaux, R., Labruyere, E., Guillén, N., & Olivo-Marin, J. C. (2011). 3-D active meshes: fast discrete deformable models for cell tracking in 3-D time-lapse microscopy. *Image Processing, IEEE Transactions on*, *20*(7), 1925-1937.
7. **[7]** Maitra, M., Gupta, R. K., & Mukherjee, M. (2012). Detection and counting of red blood cells in blood cell images using Hough transform. *International Journal of Computer Applications*, *53*(16), 18-22.
8. **[8]** Guo, D., van de Ven, A. L., & Zhou, X. (2014). Red blood cell tracking using optical flow methods. Biomedical and Health Informatics, IEEE Journal of, 18(3), 991-998.
9. **[9]** Caselles, V., Kimmel, R., & Sapiro, G. (1997). Geodesic active contours. *International journal of computer vision*, *22*(1), 61-79.