Identification of Nano-particle Absorption by Circulating Macrophage

Melisa Orta, Kimberlee Shish, Brendan Tracey

Abstract— **Detection of cells within a photoacoustic image of a slab of tissue is an inherently difficult problem. Not only are the cells themselves shape and intensity variant, but the background is bright as well. Taking a 3-D slab and reflecting it into a 2-D plane causes different orientations of the same cellular structure to emerge. This task is currently accomplished by humans, though the process is laborious and error-prone. In this paper, we describe our approach and results in trying to solve this problem using various machine learning techniques. We chose features that reflect that cells are ovular, bright, and will have some contrast with the background. We compare results obtained with different algorithms and different combinations of features, concluding that more work is necessary to obtain acceptable results.**

I. INTRODUCTION

One forthcoming branch of cancer research attempts to take advantage of carbon nanotube absorption and imaging properties. Nano-particles can be designed to be attracted to tumor cells, and can also act as contrast agents in photoacoustic imaging. With these images, researchers analyze which types of the nanoparticles should be used as "Trojan horses" to kill tumors. At Stanford, Dr. Bryan Smith is researching the convection of nano-particles from macrophages to tumor cells within tissue. In his experiments, nanoparticles are injected into the blood stream of mice with surgically implanted tumors. This analysis requires determining the percentage of nano-particles absorbed within the macrophages, which involves identifying macrophage within raster images of the nanoparticles (the cells themselves are not visible; only nanoparticles exhibit fluorescence).

In this paper, we address the automation of finding cellular structures in the raster images of the nanoparticles. Currently, trained scientists do this by hand, which is not only tedious, but also causes bottlenecks in project progress. This problem is inherently difficult in that even humans must be trained to correctly identify cells from images, and even then they only agree to ~90% accuracy. Therefore, obtaining this level of accuracy would be sufficient for automation.

II. PREPROCESSING TRAINING DATA

We were provided a single hand-marked image with boxes around regions where there are cells, and we have used this image as the "right" answer (see milestone for full image, or zoom view in Figure X). At the beginning, we trained on 40 regions of interest (25 cells, 15 not cells) provided for us, given in the form of pixel coordinates of these regions. We developed features which made the training set clearly linearly separable, but when classifying the whole image it predicted many more cellular regions than the hand marked image. We needed a better data set, and so we hand classified the top third of our image pixel by pixel using the fully marked image as a guideline. We found that our algorithms were consistently over-classifying the bottom corner of our image due to its brightness relative to the rest of the image. We hand classified the lower right corner of the image and added it to our training set. In short, the data on which we are using to train is a pixel-by-pixel marking of cell locations based on a third of the provided hand marked image. The other two thirds we use to test by eye to get a feel for how well a given algorithm is functioning

III. CHOOSING FEATURES

By examining our training data, we began to learn what identifies a region as a cell. The region has to look roughly circular or ovular, it has to be sufficiently bright, and it has to have a high contrast between it and its surroundings. Regions of high intensity but low contrast are more likely to be conglomerations of nanoparticles within the tissue rather than the macrophage, and regions of high contrast but low intensity are more likely to be noise in the image rather than an actual cellular structure.

A. *Ring Intensities*

For our first attempt at choosing features we chose a three feature set: the average intensity of the pixel, the average intensity of the 8 pixels surrounding the pixel and the average intensity of 16 pixels around those 8 pixels. In order to reduce over classification in the brighter areas, we tried a few approaches to normalization (ie normalizing mean and variance, subtracting the average intensity of the surrounding 3x3 pixel region). After performing tests with this feature, we realized that there were areas of concentration of nanoparticles that were not necessarily cells but that were of equal average pixel-intensity.

B. *Edge Features*

We decided to take advantage of local gradient and shape information. The Matlab function edge.m uses the Prewitt approximation to find points where the gradient of the grayscale intensity is large. Using this edge function and some filtering to clean up noise in the edged image, we calculated the distance from each pixel to the nearest edge in each of four main directions: right, left, up, down and used the sum of these distances as a feature in order to eliminate orientation issues. An illustration of the above two features is shown in the following figure.

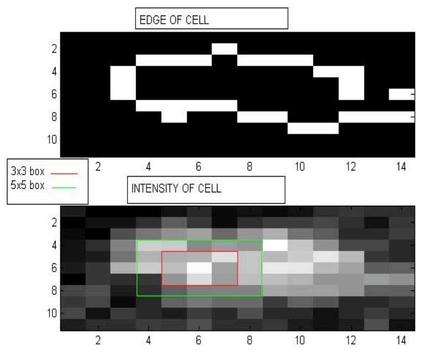

Figure 1. Example of edge and intensity features

C. *Circle Finder 1 & 2*

A circle finding algorithm was implemented to better exploit the fact that cells are roughly circular. This algorithm uses a Hough transform based on the gradient field of an image, and inputs based on the approximate size of the circles to identify the locations and radii of circular shapes. Using this method, circles were identified both in the image of the edges (obtained using edge.m, described above, circle finder 1) and in the original image (circle finder 2). A feature was assigned based on the Euclidean distance in pixels from the center of the nearest identified circle, with a threshold at 10 pixels.

D. *Morph*

For our last shape feature, we used several of Matlab's built in morphological operators. First, edge.m was run to capture the gradient information. Then fill.m and erode.m were used to fill closed edges, and cleaned up noise. Then dilate.m was used to enhance the remaining positive cell classifications. Lastly, the average intensity of the center pixel, a 3x3 box, and a 5x5 box were calculated on this altered image, and used as features. While many of the shape features alone do not find all of the cells, but it was hoped a combination of them would help classification.

E. *Individual Intensities and K-means*

When looking at components of our feature space, it was clear that it was still highly inseparable. We added one feature set of the individual intensities of each of the pixels in a 5x5 box around the test pixel, as well as the mean intensity and the variance of the intensities for a total of twenty-seven features. Lastly, to better find features in our data, we used the above feature set on all of the pixels in our image and ran k-means clustering on the data. Ordering based on mean intensities, we assigned values to each of the regions predicted by k-means, and assigned each pixel a value based on its region.

IV. Training Algorithms andFeature Selection

We used two main algorithms to test our data: Naïve Bayes, and SVM-lib [1]. Features are assigned as well as classified on a pixel by pixel basis. Our training data is very imbalanced, having only 1.5% of the pixels as cells, and is very large, comprising over 65000 pixels. As a result of these two factors and computational limitations, we reduced our training set to 70% of our total positive (is a cell) pixels, and fifteen times this number of negative pixels (is not a cell), selected at random. This is only 11.75% of our total training data. The remaining 30% of positive pixels and all remaining negative pixels are used for testing.

When training using Naïve Bayes, a k-means algorithm was used to discretize each feature space into five clusters. Then, a multinomial Naïve Bayes algorithm was run on the training data. Given the imbalance in our data set, for some of the feature spaces the algorithm would classify the data as all no's due to the high unlikelihood that any given pixel is actually a cell. As a result, for some runs we tried to optimize the decision boundary (py=1 > db* py=0) for maximum precision and recall, and minimal distance between precision and recall on the test data. After the optimal decision boundary is found, we then test our algorithm on the test data and the entire data set, and then classify the whole image. In addition, we implemented a forward search feature selection algorithm to identify important features.

We trained on SVM using both a linear and a radial basis function (RBF) kernel. Like Naïve Bayes, when training on SVM with both the linear and RBF kernels, we sometimes optimized the choice of C using a five fold k-fold testing algorithm. In the guassian kernel, gamma was chosen to be 1/n, where n is the number of features, as recommended by [1].

V. Results

We ran each of the individual features on each of the algorithms. The forward search feature selection algorithm was used to identify important feature combinations. From this, we determined that for Naïve Bayes, in descending order of importance, the top three features were 7, 2, 6. We also tried a sampling of other combinations we thought would work well together. The results are tabulated below. The linear kernel SVM results are provided to give an overview of how the features performed relative to one another, while the next two images are indicative of the relative performance of the algorithms.

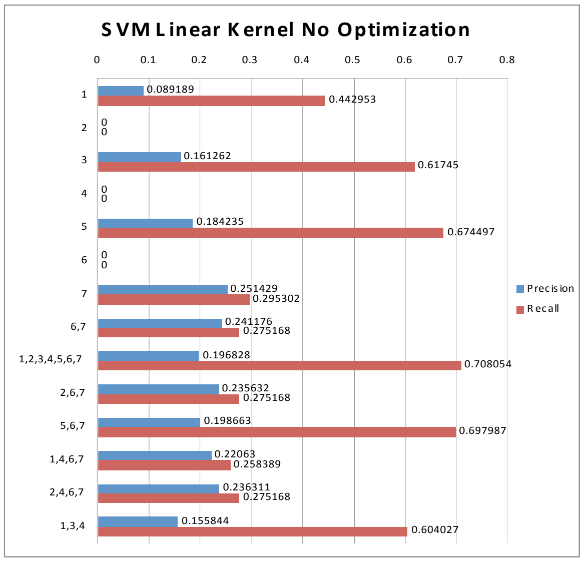

Figure 2. Results for various feature sets (1-kmeans, 2-edges, 3-ring intensities, 4-circle finder 1, 5-individual intensities, 6-circle finder 2).

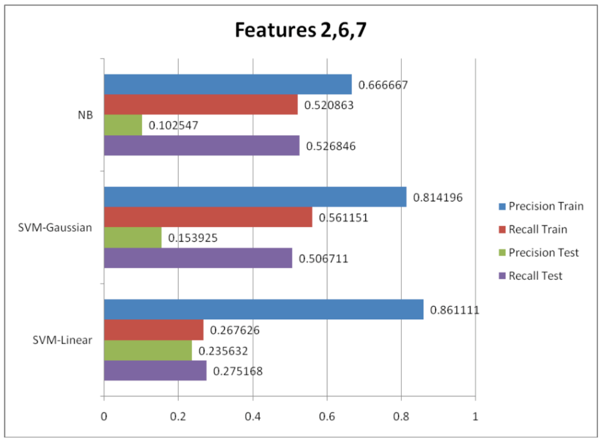

Figure 3. Comparison of algorithm performance for features found through feature selection

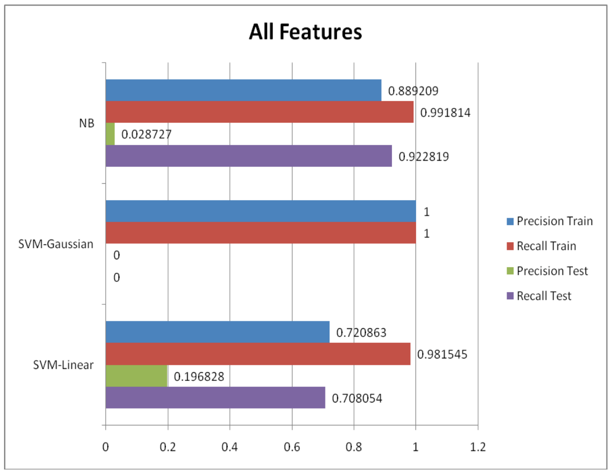

Figure 4. Comparison of algorithm performance using all features

Overall, none of the algorithms preformed much better than the others. The SVM algorithms tended to over fit our training data, performing extremely well on the training data but very poorly on the testing data. Our attempts to reduce over-fitting (through either K-fold analysis or hand picking parameters) led to one of two things: the algorithm would predict that there were no cells in the image, or the improvements would be very minimal. Naïve Bayes had less, though still significant, problems with over-fitting the data, but was not able to perform as well on the training data. Aside form the feature sets predicting no cells, attempts to optimize the decision boundary had no effect on precision and recall or actually decreased the performance on the training data due to over-fitting. In the end, the two algorithms performed roughly the same, with SVM being more accurate on the training data but over-fitting, and Naïve Bayes doing less well on the training data but having a more robust decision boundary.

VI. A Closer Look at Our Feature Space

In these results, our precision is very low, implying that our algorithms are over classifying the number of pixels which are cells. In analyzing our results we must keep in mind that classification is done on a pixel by pixel basis. If we classify a larger region of pixels as part of a cell than is actually marked in the cell, this will result in a lot of false positives. Alternatively, if we get only some of the pixels that are classified as a cell, we will have a lot of false negatives despite having accurately found the cell. In reality, the labeled boundary between cell and not cell is very imprecise, as it is impossible to tell on the scale of a pixel exactly where the cell is. Due to this uncertainty, our actual precision and recall are not completely indicative of our true precision and recall in finding regions which are cells. In the image below (Figure 5), a hand count of cells returns precision and recall values for the entire training data in the order of 0.5. However, this alone cannot completely explain our low accuracy.

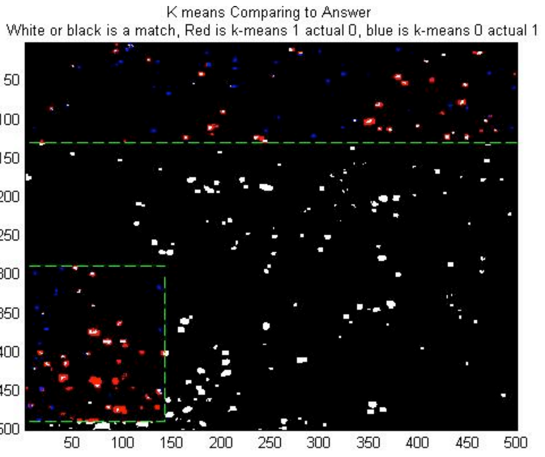

Figure 5. Illustration of misrepresentative precision and recall. Training data sections are marked off by green lines. White and black indicates train matching classification, blue indicates false negative, and red are false positive.

Each of our features has its own problems dealing with the difficulties of the classification. The edge feature has problems with the noise of the image creating edges where there should not be and blurring edges where there should be. Furthermore, the edges that are correctly placed may not match with the provided classification, creating error where there may be correct classification. Our contrast features encounter difficulties because there are some cellular regions which have high contrast, but there are many others which do not. In fact, there are many cases in which two regions that to the untrained eye look morphologically similar have different classifications, creating problems with any feature set that we could select. All of the traits around which our features were selected (shape, intensity and contrast) have high variability throughout both classification classes (see figures 6). We had hoped that combining the knowledge in each of the features would lead to a robust classification, but it appears to not be the case.

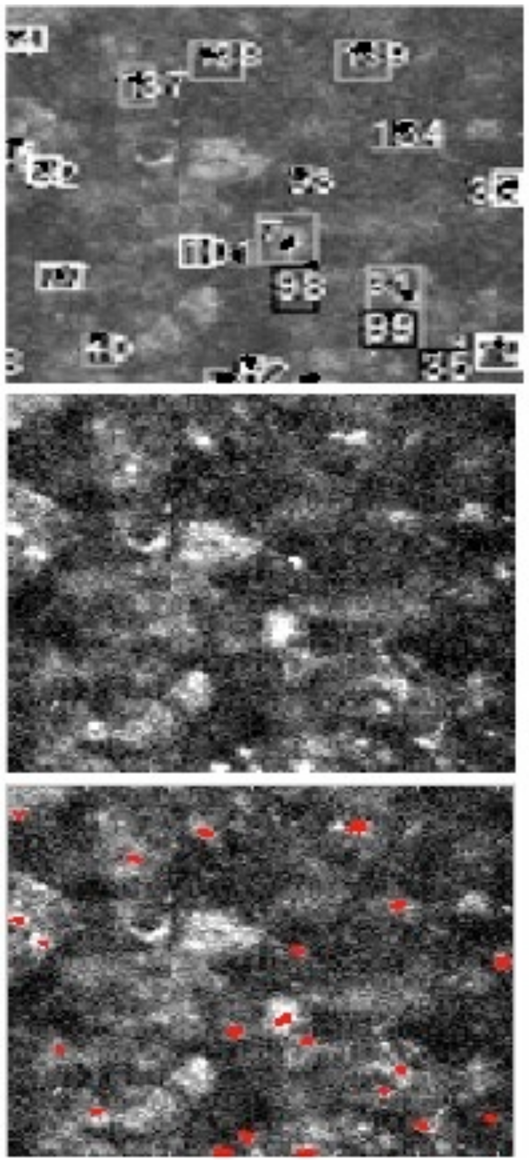

Figure 6. Illustration of photoacoustic image of nanoparticles, showing hand marked (top), unmarked (middle), and training data (bottom) examples.

VII. Future Work

One method that may help improve our classification is the CISS algorithm which deals with finding the most representative data in a data set that is very imbalanced. We propose to implement this algorithm, which iteratively trains on the poorly classified data, to improve our results.

A more fundamental shift in classification procedure would be to try to find regions that are cells rather than classifying on pixels. This approach is potentially more invariant to noise in the image and to mislabeling of cell boundaries. On the other hand, a pixel by pixel classification has the upside of being able to correctly identify a cell boundary when working correctly, and will aid in the computation of percent uptake.

Lastly, a property of the cells that we did not explore for this part of our work is that they move with respect to the other tissue. One good example of this is illustrated in the following image sequence:

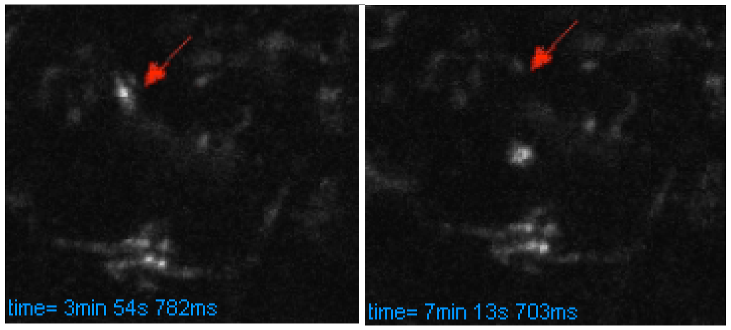

Figure 7. Screenshots of video of cell movement

In conclusion, we have generated a variety of features and applied a number of machine learning techniques to this problem with disappointing results. In order to break further ground on this problem, either more expertise will be needed to generate more precise and robust features, or a brand new method must be employed which uses time-lapse images or at the very least higher resolution images with less noise

Acknowledgment

This project could not have been completed without the help of Dr. Bryan Smith and Farchid Moussavi. Dr. Smith provided the motivation, understanding, and training data for this work. Farchid Moussavi connected us with Bryan, suggested the use of the morph features, gave feedback on our ideas, and was the source of much encouragement.

References

[1] Chang, Chih-Chung, LIBSVM-A Library for Support Vector Machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

[2] G. X. Long, W.L. Cleveland, Y.L. Yao, Automatic detection of unstained viable cells in bright field images using a support vector machine with an improved training procedure, Computers in Biology and Medicine 36 (2006) 339–362.