THESIS

A COMPUTER VISION APPROACH TO CLASSIFICATION OF CIRCULATING TUMOR CELLS

Sumbitted by

David Hopkins

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2013

Master's Committee:

Advisor: Michael Kirby

Chris Peterson
Geof Givens

ABSTRACT

A COMPUTER VISION APPROACH TO
CLASSIFICATION OF CIRCULATING TUMOR CELLS

Current research into the detection and characterization of circulating tumor cells (CTCs) in the bloodstream can be used to assess the threat to a potential cancer victim. We have determined specific goals to further the understanding of these cells. 1) Full automation of an algorithm to overcome the current methods challenges of being labor-intensive and time-consuming, 2) Detection of single CTC cells amongst several million white blood cells given digital imagery of a panel of blood, and 3) Objective classification of white blood cells, CTCs, and potential sub-types.

We demonstrate in this paper the developed theory, code and implementation necessary for addressing these goals using mathematics and computer vision techniques. These include: 1) Formation of a completely data-driven methodology, and 2) Use of Bag of Features computer vision technique coupled with custom-built pixel-centric feature descriptors, 3) Use of clustering techniques such as K-means and Hierarchical clustering as a robust classification method to glean insights into cell characteristics.

To objectively determine the adequacy of our approach, we test our algorithm against three benchmarks: sensitivity/specificity in classification, nontrivial event detection, and rotational invariance. The algorithm performed well with the first two, and we provide possible modifications to improve performance on the third.

The results of the sensitivity and specificity benchmark are important. The unfiltered data we used to test our algorithm were images of blood panels containing 44,914 WBCs and 39 CTCs. The algorithm classified 67.5 percent of CTCs into an outlier cluster containing only 300 cells. A simple modification brought the classification rate up to 80 percent of total CTCs. This modification brings the cluster count to only 400 cells. This is a significant reduction in cells a pathologist would sort through as it is only .9 percent of the total data.

ACKNOWLEDGEMENTS

To Dr. Kirby, Ms. Emerson,

and my Co-advisors Dr. Peterson and Dr. Givens,

Your day-in and day-out attitude and commitment has made this research a joy.

To Dr. O'Hara,

Thank you for your time and input to make this thesis better.

To the researchers and pathologists at the Scripps Research Institute,

Your collaboration, enthusiasm and commitment to this research has been invaluable.

To my family, mentors and my amazing wife,

I have been blessed more than you know

by your presence and encouragement through this all.

DEDICATION

In memory of Pam Edrich.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Problem motivation and the Scripps Research Insititue

The Kuhn Lab at the Scripps Research Institute in La Jolla, California has a mission of furthering research, creating deeper understanding, and utilizing the expertise of scientists, clinicians, and patients to "determine the biological and clinical significance of circulating tumor cells (CTCs)" [16].

Their research, headed by Peter Kuhn, Ph.D.[1], involves the isolation and identification of CTCs in a blood stream. This task proves enormously difficult because of the low concentration which CTCs exist in a patient's peripheral blood (1 to 10 CTCs per mL of whole blood compared to several million white blood cells) [38]. Their efforts have created a reliable way to detect and characterize CTCs. However, a classification structure based on low-level features of cells and labeling cells objectively can assist discovery and knowledge of cell characterization. A low-level feature approach, or a pixel-by-pixel approach, to examine cell features can also be used alongside current techniques to improve classification results.

In their own words, they see that "detection and characterization of these cells is a promising method for both diagnosis and clinical management of cancer patients as well as monitoring treatment" [16]. It was with this underlying motivation that we strove to produce the research in this thesis.

1.2 Problem definition and solution presented

The problem at hand is the detection, characterization, and classification of circulating tumor cells (CTCs).

The solution proposed in this thesis has the following specifics: the extraction of low-level key features from cells, the characterization of cells from these features, and the use of unsupervised clustering algorithms, and finally the construction of a decision tree with objective rules to classify cells.

[1]Director, Scripps-PARC Institute for Advanced Biomedical Sciences. PKuhn (at) scripps.edu

With a mathematical framework guiding this process, we hope cancer researchers will benefit in new ways of understanding CTC classification from our work. An automated classification approach could save time for pathologists and researchers. The current manual method is a bottleneck in the cancer screening process. The new approach could also give novel insights into the distinctive features and subtypes of these cells.

1.3 Summary of key findings

To objectively determine the adequacy of our approach, we test our algorithm against three benchmarks introduced in Section 2.5: sensitivity/specificity in classification, nontrivial event detection, and rotational invariance. We performed well in all but the last of these benchmarks. Future modifications are suggested in Section 7.4.1 to overcome this issue.

The results of the first benchmark are important. The unfiltered data we used to test our algorithm were images of blood panels containing 44,914 WBCs and 39 CTCs. The algorithm classified 67.5 percent of CTCs into an outlier cluster containing only only 300 cells. A simple modification is added in Section 7.4.3 which brings the classification rate up to 80 percent. This modification brings the cluster count to only 400 cells.

An example of an end result is given in Figure 1 below. This result shows detection and classification of both WBCs and CTCs in a blood panel.

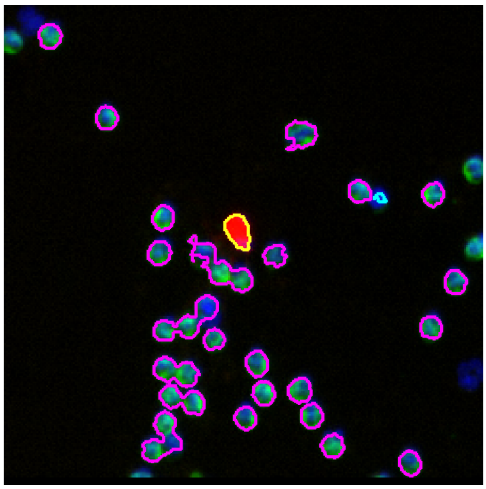


Figure 1: Circulating tumor cell (middle cell in red) correctly distinguished and classified by computer vision. (automated drawing of brightly colored classification borders: yellow, purple and teal.)

1.4 Layout of thesis

The paper introduces the necessary background for CTC imagery in Section 2, presents the computer vision technique Bag of Features in Section 3, and explains clustering algorithms in Section 4. Then, we use those concepts to build our full methodology in Section 5. In Section 6, we discuss the results of our algorithm on real data provided by the Scripps Research Institute. In Section 7, we examine the method in general by discussing strengths, weaknesses, added value and future modifications.

2 Background

2.1 Current detection method at the Scripps Institute

2.1.1 Panel specifics

The researchers at the Scripps Research Institute draw blood from hundreds of patients, both with and without cancer. They centrifuge the blood to separate the white blood and circulating tumor cells from the plasma and the red blood cells. After putting the CTCs and WBCs on a glass platelet, they then add fluorescing chemical stains which color the platelet. These chemicals are DAPI, Cytokeratin, and CD45. Their respective markings are blue nuclear stain, red CTC antibody stain, and green WBC antibody stain [17]. Once the cells are dyed, researchers take high-resolution monochromatic images of the panels.

The data we received at the beginning of the analysis were procured from panels of blood digitally imaged at 10× resolution. These panels contained thousands of white blood cells and around 40 CTCs. The second set of data consisted of two hundred cropped images centered around CTCs digitally imaged at 40× resolution. The relationship between a panel and a cropped image is in the zoom level of the image. A panel shows one to two thousand cells, where cropped images are zoomed in around specific CTCs within a panel. The cropped images show fifteen to forty cells.

Each image is taken at 1362×1004 pixels (for the panels) and 250×250 pixels (for the cropped images).

Figure 2 below is an example image displaying the various stains on a panel.

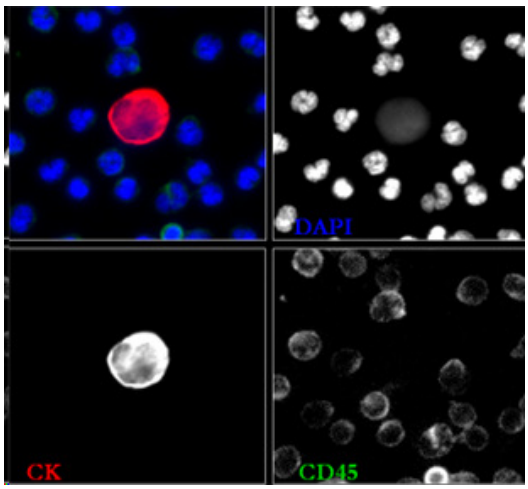


Figure 2: The researchers add fluorescing chemical stains which color the white blood and circulating tumor cells on the platelet. These chemicals are DAPI, Cytokeratin, and CD45. Their respective markings are blue nuclear stain, red CTC antibody stain, and green WBC antibody stain [17]. Image from kuhn.scripps.edu.

2.1.2 Current methods for CTC classification

The current methodology in locating CTCs involves searching the panels for areas with a high concentration of expression from both the Cytokeratin and DAPI stains. Areas with high concentration in both of these are hand-checked by a pathologist to determine false positives from potential CTCs. The potential CTCs are then hand-checked by another pathologist for verification.

Even though CTCs exist in low concentrations ($1 - 10$ CTCs per mL of whole blood as compared to several million white blood cells (WBCs) [38]), the current approach is very accurate in detecting the presence of CTCs.

2.1.3 Challenges in current method

The first challenge to CTC classification is that the red dye occasionally shows up without the presence of Cytokeratin.This leads to what is called 'red noise' and, if not accounted for, can influence CTC detection as false positives. An example of this is seen in the bottom left corner of Figure 3 below. A CTC detection method should err on the side of having more

false positives rather than more missed detections of CTCs. With such rare occurrences, any information possible about potential CTCs is crucial to know. The human interaction with a detection algorithm would be interpreting results and filtering out any false positives.

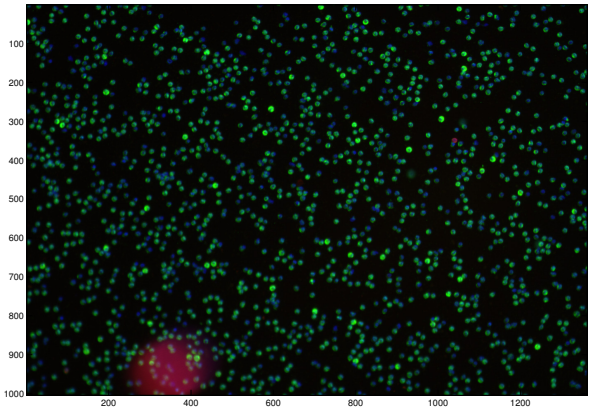


Figure 3: Imperfections in adding the fluorescing chemicals may sometimes smudge the panels, leaving 'red noise'.

Current methods are also prone to being subjective. Formal attempts to qualitatively define a morphological model for CTCs has taken many different paths and are largely subject to inter-operator variation [3]. In fact, a recent study showed that the "absolute number of cells found in patients and normal donors varied by more than a decade between different morphological definitions" [8].

Additional challenges include the fact that white blood cells are unlabeled, that cells may overlap within images, and the sheer size of the problem: Each panel contains somewhere between 1,500 and 2,000 cells.

Overall, the current approach is extremely labor-intensive and time-consuming. However, automated computer vision techniques can help overcome many of these issues.

2.2 Other approaches to cell classification

The challenges facing CTC detection and subclassification lie in parallel with cell classification across other areas of medical imaging. To overcome the challenges of being labor-

intensive, time-consuming and subjective, several types of automated approaches using Computer Aided Diagnostic (CAD) systems have been developed.

In general, the CAD methodologies involve hand-picking a high-level set of features for training algorithms. These specially tailored feature sets are constructed from a variety of statistics and morphological features about cells. Within the area of human epithelial cells (HEp-2) for example, Cordelli et al. [4] build descriptors from image energy, mean, entropy and texture. On the other hand, Strandmark et al. [34] use area of the convex hull, eccentricity, Euler number and perimeter. The total number of features in the descriptors of these approaches are 15 and 966 respectively.

Although HEp-2 cell imagery is different from CTC imagery, HEp-2 cells are used as a standard set in cell classification algorithms. There are enough key similarities to warrant comparisons of our approach with different approaches and the state of the art techniques. The two types of cellular imagery are similar in that they are acquired by means of Indirect ImmunoFluorescence and require building feature descriptors to train CAD systems.

To best compare classification results in cell imagery, the International Conference on Pattern Recognition (ICPR) 2012 [1] hosted a classification contest on HEp-2 cells. HEp-2 cells have a standard set of six classification labels for any cell: homogenous, coarse-speckled, fine-specked, nucleolar, centromere, and cytoplasmatic [34]. The training data set for the ICPR contest consists of 721 individually segmented and labeled cells. The test data set, 736 cells, is not yet publicly available [1].

Even while using the same training set, classification rates across various approaches are hard to compare because authors have used different methods of validation. For example, many authors use ten-fold cross-validation [37],[4],[25], while one [34] uses leave-one-out cross-validation.

The contest winner and the state-of-the-art method in classification for HEp-2 cells is Cordelli et al. with 79.3% accuracy (ten-fold cross-validation error rate) [4].

In all of these classification procedures, a set of features to describe cells is needed. The clearest distinction our work has from all but the classification approach by Wiliem et al.

is the use of data-driven low-level features rather than hand-picked high-level features. We describe in Section 5.2 precisely how we create these features as part of the computer vision technique known as Bag of Features.

Some of the various approaches in literature are shown in Table 1 below. We position ourselves within this taxonomy for comparison.

Table 1: Current methodologies in cell classification
Most information from columns one through three gleaned from overview table in [37]. Information in columns four and five found from citations listed.

Authors	Feature descriptors	Classifier	Private data classification rates	ICPR contest winner's rates
Strandmark et al. [34]	Morphological, statistical, textural	Random Forest	96.1%	- -
Wiliem et al. [37]	Dual-Region codebook-based, Discrete Cosine Transform	Nearest Convex Hull	95.5%	- -
Hiemann et al. [11]	Structural, textural	LogisticModel Tree	94.6%	- -
Elbischger et al. [7]	Statistical, shape, textural	Nearest Neighbor	90.25%	- -
Cordelli et al. [4]	Statistical, textural, morphological	Adaboost	85%	79.3%
Hsieh et al. [12]	Statistical, textural	Learning Vector Quantization	80.3%	- -
Soda et. al. [31]	Textural	Multi Expert System	75.9%	- -
Hopkins (This Thesis)	Single-Region codebook-based, Modified Scale Invariant Feature Transform	Hierarchical, Nearest Neighbor	80%	- -

2.3 Using computer vision technique: Bag of Features

The goal of Bag of Features is to generically characterize images using a dictionary of low-level features. The simplicity and impressive results of Bag of Features may explain why it has grown so popular in the past decade [24]. It has given successful results in applications such as image classification, video search [14] and texture recognition [23]. In the area of scene categorization, a Bag of Features method exceeded the performance of the state-of-the-art on the Caltech-101 database [19]. This method has also set benchmarks for image classification and standards in image retrieval [24].

Among these applications, Bag of Features has also been used for classifying Human Epithelial type 2 cells from indirect immunofluoresence images [37]. The methodology by Wiliem et al. (shown in Table 1) uses Discrete Cosine Transform (DCT) descriptors to describe cells' low-level features. This kind of low-level feature descriptor is calculated by sectioning a cell into small partitions and creating a vector of features from within each partition. For each partition, posterior probabilities of the vectors are calculated and the results are placed in a histogram. Then, all of these histograms are averaged to give a histogram representing an entire cell [28].

A simpler version of these descriptors is to only use a single histogram. The within-partition features are found using the Scale Invariant Feature Transform (SIFT) descriptor proposed by Lowe [20]. These descriptors are invariant to scaling, orientation and partial changes in illumination. These descriptors give a spatially-orderless approach for image representation [19]. The feature descriptors we build are modeled after these and discussed in Section 5.2.

2.4 A motivation for cluster analysis

For our specific task at hand, we seek more than solely identifying circulating tumor cells from white blood cells. We seek to use computer vision to, ideally, aid clinicians in discovering new CTC subtypes. Because this task requires dealing with new classifications, we enter

into the area of structuring massive unlabeled data sets. This task in unsupervised learning presents one key challenge: that there is no error or reward signal to evaluate a potential solution.

A key technique in unsupervised learning is to use clustering algorithms. Clustering involves finding relations and structure amongst the data. Clustering algorithms have a native strength that can overcome the challenge of unsupervised learning. This is the fact that they do not require labels on the data to reveal relationships and structure. For example, one may cluster all cells with certain feature descriptors. Also, one may cluster cells based on the frequency of certain features observed. In both of these examples, the clustering did not depend on whether the cell was a CTC or WBC.

In this thesis, we use two clustering techniques which aid each other in classification of cells and setting the path to discover cell subtypes. These two techniques are the K-means algorithm (flat clustering) and the hierarchical descent algorithm (hierarchical). We describe these in detail in Section 4.1 and Section 4.2, respectively.

Performance for the K-means clustering techniques is measured by accuracy (the sum of squared error (SSE)) and parsimony (the total number of clusters used). The SSE and the number of clusters used have an inverse relationship. As we increase the number of clusters, the clustering results get more accurate (SSE decreases). So the performance of the K-means approach depends on an optimal balance between these two. We discuss this fully and give an objective way to find this balance in Section 4.1.1.

Finally, performance of hierarchical clustering algorithms is done by the F-measure [33]. This measure is defined as a harmonic mean of precision and recall [29], where precision is related to specificity or the fraction of results that are relevant and recall is sensitivity or the fraction of relevant results returned. For further clarification, maximum precision means no false positives and maximum recall means no false negatives.

In our case, have labels for our training set on whether an event is a CTC, WBC or red noise. So, we report the precision and recall of the clustering algorithm for finding CTCs. We do not report the F-measure because this assumes that specificity and sensitivity are equally

important. And we have discussed that sensitivity is more important. We also measure the performance of hierarchical clustering by seeing which measure of distance achieves the greatest diversity in the overall structure. From there, we can locate any particularly insightful clusters displaying precision and recall of CTCs. We discuss this further in Section 5.5.

2.5 Tests to benchmark performance

We propose three tests to benchmark the performance of a detection algorithm. Ideally, we seek to greatly reduce, if not eliminate, the human effort required to identify CTCs. So, any well-developed method must severely limit human interaction during the bulk of the procedure.

An algorithm will operate on a data cube representing the composite image of a panel or cropped image. The data cube has length and width equal to the length and width (in pixels) of its associated panel or cropped image. And it has a height of three, associated with the layers of the composite image. The layers of this composite image are the Cytokeratin, CD45, and DAPI images. These individual layers correspond to the red, green and blue layers, respectively, of the composite image.

The tests are as follows:

1. **Sensitive and specific classification of CTCs**

   When we input a panel into the algorithm, the output should be similar events grouped into various 'buckets.' Ideally, these buckets will have some degree of interpretability and be capable of differentiating between white blood cells, circulating tumor cells and noise. A successful outcome would be if the majority of events in a bucket belonged to a certain cell type.

2. **Non-trivial event detection**

   Recall that the presence of Cytokeratin is a key feature in distinguishing a CTC from a white blood cell. Further, recall that a red stain may also indicate the possibility of a red-noise event on a panel. With this in mind, it is important to assess an algorithm's

specificity in classifying these 'red events'. A test which can prove to be more than a 'red' classifier will be deemed successful.

3. **Rotational invariance**

   The characterization of a cell should be independent of orientation in the image. Because of this, a cell classified in Bucket A should continue to be classified into Bucket A when rotated.

We review these tests and use them to assess our proposed algorithm in the results and in discussion Section 6.

3 Computer vision technique: Bag of Features

The field of computer vision concerns the analysis and understanding of real digital image data. The idea that a computer can 'understand' an image involves the "disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory" [9]. To attempt to address the challenges of the CTC classification problem, we will combine the benefits of a data-driven methodology and power of computer vision techniques.

In this section, we give an overview of the computer vision technique called Bag of Features [24]. This technique was initially introduced to identify the object content of natural images [5]. We will detail each component necessary to understand the algorithm. For this, we formulate image gradients, introduce feature descriptors, then describe term vectors.

The core of the Bag of Features method involves developing a dictionary of possible features in an image. Then, one simply counts the frequency of feature-similar occurrences in a new image. The vector of counts, the term vector, holds adequate information for image characterization.

To introduce this, we turn to the idea of "term vectors" as applied to text mining. Text mining is the science of describing and analyzing text document information amongst several documents. To describe each document, an object called a term vector is made. A primitive, but effective way to describe a document is to build a vector the length of the dictionary whose elements correspond one-to-one with each word. The term vector would house frequency information for how often a particular word in a document is used. Naturally, the term vectors would be quite sparse, as most words go unused, but a different dictionary could be created containing only the important words to consider in the document analysis.

Two documents of entirely different genres can be compared by observing the relationship between their respective term vectors. For example, suppose document A relates to the field of biology. It uses the words: "cell", "mitochondria", "protein" and "enzyme" a relatively high percent of the time. For each occurrence, term vector A has a 1 placed in

the element corresponding to that word. Suppose document B relates to music composition and frequently uses the words: "Mozart", "Strauss", and "Beethoven". Term vector B is updated likewise. The vectors from documents A and B would hardly resemble each other. The two respective term vectors would have values in completely different elements. Using the Cosine metric [30] the distance between the two term vectors would be very large.

The analogy is drawn here between the document example and CTC classification. Each event is considered a separate document and is described with a term vector. But unlike documents, whose existence presupposes a written language, one may wonder how to build a term vector for images without the use of a dictionary. In our case, the dictionary must be built from words relating to and created from the feature descriptors. The words in our dictionary are formed in our method by clustering the entire set of feature descriptors then building a one-to-one correspondence between cluster centroids and element positions in a term vector. The idea is described with pictures for our specific problem in the Section 5.4.

A basic schematic of the steps involved in Bag of Features as described in [5] are given below. In the following subsections, we elaborate on the words in bold.

1. Describe section of an image using **image gradient** information
2. Build a **feature descriptor** from this information
3. Assign feature descriptor to a set of predetermined clusters (the dictionary)
4. Build an image **term vector**, which holds the occurrence frequencies assigned to each cluster
5. Cluster the image term vector to determine image classification.

3.1 Image gradients

The Bag of Features algorithm requires the computation of image gradient information. Here, we describe image gradients and a method one can use to calculate them.

Let I be a gray-scale image. That is, each element in I is a value in the range $\{0, 255\}$. Consider the element I_{ij} and its nearby elements

$$\begin{pmatrix} & \vdots & \vdots & \\ \dots & I_{i,j} & I_{i,j+1} & \dots \\ \dots & I_{i+1,j} & I_{i+1,j+1} & \dots \\ & \vdots & \vdots & \end{pmatrix}.$$

To compute the first order derivative in the horizontal direction, we use the following formula

$$(I_{ij})_x = \frac{\delta I_{ij}}{\delta x} = 1/2((I_{i,j+1} - I_{i,j}) + (I_{i+1,j+1} - I_{i+1,j})).$$

Similarly, to compute the first derivative in the vertical direction, we use

$$(I_{ij})_y = \frac{\delta I_{ij}}{\delta y} = 1/2((I_{i+1,j} - I_{i,j}) + (I_{i+1,j+1} - I_{i,j+1})).$$

Putting these together, we get the gradient of I defined as

$$\nabla I = \begin{pmatrix} I_x & I_y \end{pmatrix}.$$

Along with a vertical and horizontal direction, we'd also like to keep track of the gradients magnitude and direction. A key property of the image gradient is that at any given pixel, it points in the direction of greatest intensity change. The equations for the magnitudes and directions are simply a transformation to polar coordinates performed as

$$||\nabla I|| = \sqrt{I_x^2 + I_y^2}$$

and

$$\tan(\theta) = \frac{I_y}{I_x}$$

respectively.

The next step is to use this gradient information to determine a feature descriptor for all pixels inside an event.

3.2 Feature descriptors

The main purpose of a feature descriptor is to describe local curvature and intensity information in an image.

The key property of a feature descriptor is repeatability [5].Suppose you have two originally identical images. Then one of the images has alterations applied to it. This could be that the image is rotated, resized, changed illumination, or any combination.

A feature descriptor that has the property of repeatability is one that can describe different views of the same object in a robust way, regardless of image alterations. That is, when two views of the same object are compared, you would prefer their feature descriptor to also be nearly identical.

Another desirable property for feature descriptors is detailed information to describe image segments. This can be accomplished by feature descriptors with a large number of components. A large feature descriptor vector has "rich and potentially more discriminative representation" [5] than alternatives methods to describe features such as steered Gaussian derivatives or differential invariants [5]. For example, the well-performing Scale Invariant Feature Transform (SIFT) descriptor has length 128×1 [20] while the Gaussian derivatives and the differential invariants have lengths 12 *to* 16×1 [5].

A unique, pixel-centric feature descriptor was constructed with these desirable properties in mind specifically for the CTC detection algorithm. The details of this feature descriptors are described in full mathematical detail in Section 5.2.1.

3.3 Term vectors

Having a feature descriptor that describes information about a pixel in an event, in our case a cell, is the first step in actively describing the cell information in a cohesive way. We'd

like some way to compare cells together based on their feature descriptors. This calls for building vectors that describe each cell and that compare readily with other cells' vectors.

To accomplish this, we compute a term vector for each cell. Each entry in a term vector is associated with a particular type of feature descriptor. These main types of feature descriptors are determined by the fact that they are the cluster centers found by using nearest neighbor clustering algorithm. We choose an adequate-sized set for these main types by using clustering principles discussed in Section 4.1.1.

Let's walk through how to build a term vector for a given event. Let us locate a pixel then compute its feature descriptor. Next, we locate its most similar feature descriptor type (its nearest cluster center), then we find that type's associated position in the term vectors. We then just increase that position entry by one for the occurrence of that feature descriptor. After doing this for all the pixels in an event, we have constructed the associated term vector for that event.

After constructing the term vectors for various events, we seek to gain insights on their overarching structure. We will develop supervised and unsupervised learning techniques to learn to classify new unknown events. To accomplish this task, we turn to the next section on Clustering algorithms.

4 Clustering algorithms

4.1 K-means

Given raw data in a measurement space, it is often advantageous to compress this data into a feature space by finding relevant clusters. These clusters and their cluster centers help to simplify data interpretation. The K-means algorithm is an iterative, centroid-based clustering technique which minimizes the distance between data points and their assigned cluster center.

As described in [27][6], the objective function for the minimization is

$$\min_{\mu_1,\mu_2,\ldots,\mu_K} \sum_{k=1}^{K} \sum_{x \in X_k} d(x, \mu_k)^2.$$

where

x is a data point

K is the total number of clusters

X_k is the kth cluster

μ_k is the kth cluster centroid

$d(x, \mu_k)$ is the distance between data point x and cluster centroid μ_k.

4.1.1 Distortion error for preselecting K

The K-means algorithm and its results depend on the significant value of K, which must be decided *a priori*. If structural properties about the data are known beforehand, then K may be decided with that in mind. However, when this isn't the case, procedures for selecting K are far more ambiguous. The choice of K has been a subject of research throughout the years. For an $m \times n$ data matrix, methods methods developed have ranged from simple rules of thumb: $(k \approx \sqrt{n/2})$ [21] to statistical-based information criterion approaches (Akaike information criterion) [10]. In all these methods, the choice of K seeks to balance parsimony (smaller Ks are preferred to larger Ks) and accuracy (a minimal total sum of squared errors

is preferred).

Amongst the feature descriptors in our dataset, there does not yet exist a sufficient understanding of structural information to determine K beforehand. Therefore, we choose to examine the distortion error curve, also called the "elbow method", to find the value of K. This method balances parsimony and accuracy by plotting the two variables, K and SSE, against each other. Then, it considers which values of K provide relatively small reductions in SSE. The key idea driving this method is to build a model as simple but accurate as possible. One chooses the number of centroids so that the addition of another negligibly improves the accuracy of the clustering.

Figure 4 below illustrates the resulting curve from running the K-means algorithm several times with various values chosen for K. The bend in the curve is known as the 'elbow' and is the namesake of the methodology. It is the value of K, circled in red, at the elbow that is the value suggested for use.

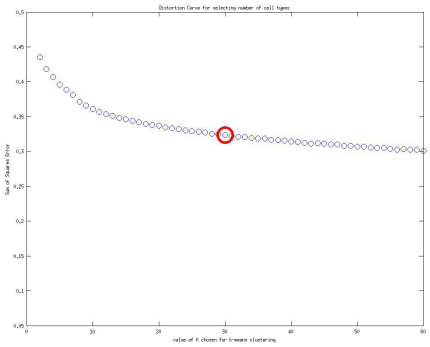


Figure 4: The elbow plot shown above helps identify the number of clusters to select beforehand for the K-means algorithm. The x-axis marks values of K. The y-axis marks values of the sum of squared error. The curve is shaped like an 'elbow' and the suggested value of K is at the elbows bend.

4.1.2 Cosine metric

Another subjective aspect of the K-means algorithm is that of choosing the distance measure to use in the minimization function. That is, the choice of $d(\vec{x}, \mu)$ is crucial in determining clusterings. In the use of text and document analysis, the Cosine-similarity metric is "the

baseline for most similarity studies" [26].

The Cosine-similarity measure is defined as

$$d(\vec{x}, \vec{y}) = 1 - \theta$$

where

$$\cos(\theta) = \frac{\vec{x}^T \vec{y}}{||\vec{x}||||\vec{y}||}.$$

A useful property is that this measure allows for variations in pixel intensity. Because this measure is widely used in text analysis, it is promising for clustering in the use of our feature descriptor vectors and term vectors. This is, of course, not left to speculation and the analysis of this measure as compared with the Euclidean measure for clustering term vectors is shown in Section 5.5.

4.2 Hierarchical clustering

K-means clustering is an algorithm known to be a type of flat clustering. That is, the algorithm determines the entire set of clusters simultaneously. Although the K-means algorithm is efficient and we've presented a way to overcome the problem of pre-specifying the number of clusters to use, flat clustering techniques can not aid in understanding the structure of clustered data.

For such understanding, we turn to the second archetype of clustering algorithms, hierarchical clustering. The structure of the algorithm has, at its roots, a pairwise distance matrix calculated between all points.

A basic procedure for the technique is as follows [15]:

Let $\boldsymbol{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ be a set of data vectors we wish to cluster.

Let $P(\boldsymbol{X})$ be the pairwise distance matrix.

1. Assign each vector to its own cluster. The cluster count is now at N.

2. Link the closest pair of clusters together. The cluster count is now at $N - 1$.

3. Recalculate a Pairwise distance matrix between the old clusters and the new cluster.

4. Repeat steps 2 and 3 until there is only one cluster left.

In this process, we have the freedom to choose which distance measure to utilize. We will choose to use the Cosine measure for our distance measure. This is to remain consistent with literature [26] and the K-means part of our algorithm as discussed in Section 4.1. Secondly, the way that clusters are linked together may be defined in a myriad of ways. We chose to use the unweighted average distance method [32] to compute the between-cluster distances. This method was chosen because it was compatible with the Cosine measure, unlike most other methods, such as the Centroid, Ward, and Weighted Center of Mass Cluster distances, which are only appropriate when using the Euclidean distance.

The unweighted distance measure between two clusters of vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ is

$$d(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{|\boldsymbol{X}||\boldsymbol{Y}|} \sum_{\vec{x} \in \boldsymbol{X}} \sum_{\vec{y} \in \boldsymbol{Y}} d(\vec{x}, \vec{y})$$

where $d(\vec{x}, \vec{y})$ is the cosine distance metric.

This may be interpreted as being the mean distance between all elements of each cluster. "This differs from the linkage methods in that it uses information about all pairs of distances, not just the nearest or the furthest. For this reason, it is usually preferred to the single and complete linkage methods for cluster analysis" [22].

4.2.1 Determining number of clusters from structure: the dendrogram

The results of the hierarchical clustering algorithm can be displayed using a chart called a dendrogram. This chart reveals how the clusters connect to one another and their parent clusters. It also shows the between-cluster distances by the lines connecting the various cluster nodes. The vertical height of each of these lines is the relative distance between nodes. In other words, "When the connecting lines are a little higher in value the series are less similar to one another; the relative distances away from each other are greater" [18].

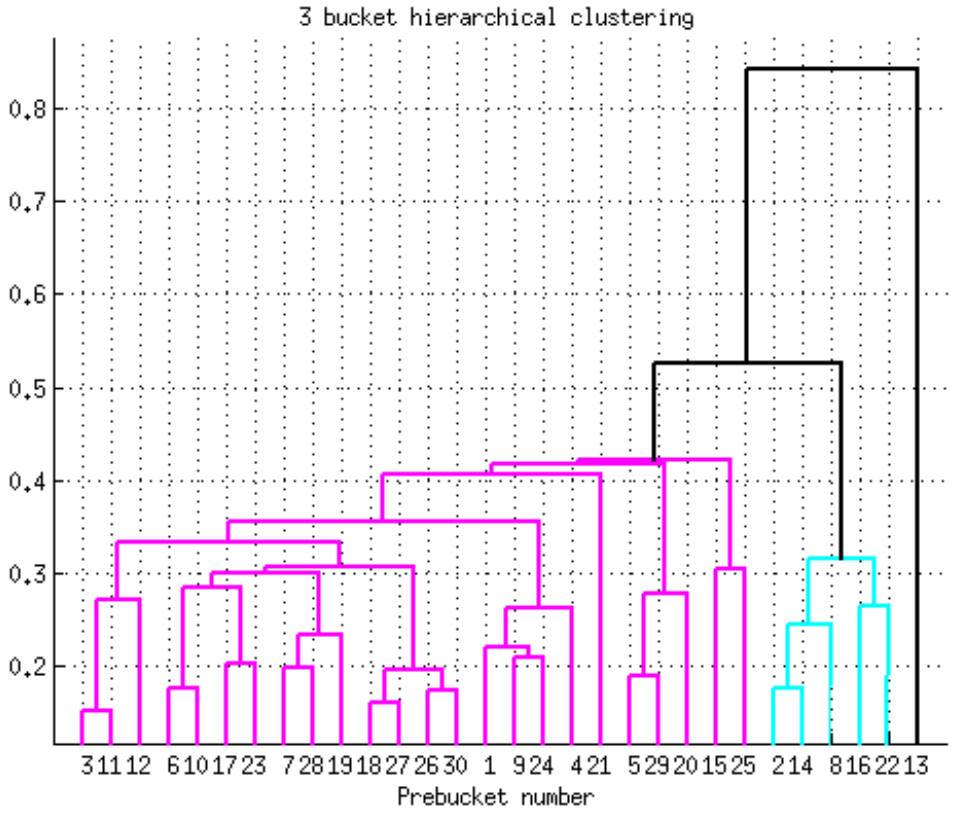


Figure 5: This is an example dendrogram. The numbers on the x-axis are the indices of 30 data points. The branches linking them display the distances between the points. It is important to note that the distances between points can only be interpreted by the height of their connecting branches and not by their x-axis distance from one to another. Following the rule of thumb given above in Section 4.2.1, the dendrogram tree was cut at a y-value of .45. This leads to three clusters: the heavily populated pink cluster, the light blue cluster and the anomaly black cluster (the isolated branch on the right is associated with data point 13.)

To determine the appropriate number of clusters, one tries to identify where the relative distances between nodes drastically change from near to far. From the structure information shown in the dendrogram, this may be interpreted as finding the horizontal spot where most of the vertical lines change from being long to being short. Though this provides ample structuring information, we note that with the dendrogram alone we cannot determine exactly how similar (or dissimilar) one series is to another. We can only see how relatively close one series is to the cluster formed by the others in the comparison set.

5 Methodology for CTC subclassification

The main purpose of this section is to walk step-by-step through the proposed algorithm.

An overview of algorithm is as follows:

1. Segment cells from panel.
2. Compute 'bottom up' feature descriptor for each pixel in an event.
3. K-means cluster feature descriptors to build a visual dictionary.
4. Input new panels. Repeat steps 1 and 2 above.
5. Construct term vectors for new events with respect to the visual dictionary.
6. K-means cluster these term events into 'prebuckets' for efficiency.
7. Hierarchical cluster the prebuckets to form 'buckets'. These describe the main classification of events and their relative relation to each other.

5.1 Cell segmentation

Within the CTC images, it is necessary to mark events of interest (cells, red noise) from the black background. We accomplish this by means of a standard edge detector. MATLAB's built-in edge detectors converts an RGB composite image to a gray-scale image. Then, it converts the image into a binary image (black or white image) based on thresholding the gray-scale values. If the gray-scale value of a given pixel is greater than the threshold, set the pixel value to 1; if not, set the pixel value to 0, corresponding to white or black, respectively. The edge detector then determines whether two pixels are connected if their values are the same and connects the two. We then trace the edges of the connected regions. When the threshold parameter is set appropriately, our edge detection yields adequate results to continue to the next step in our algorithm.

Figure 6 and Figure 7 below show the results of the edge detection on a cropped image and a full panel.

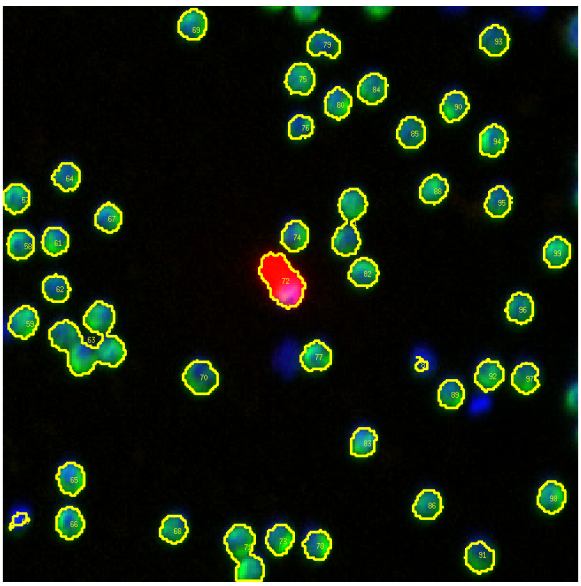


Figure 6: Edge detection shown on panel cropping around a CTC. The numbering of the cells is added to give each event a unique identifier for classification results in the testing phase.

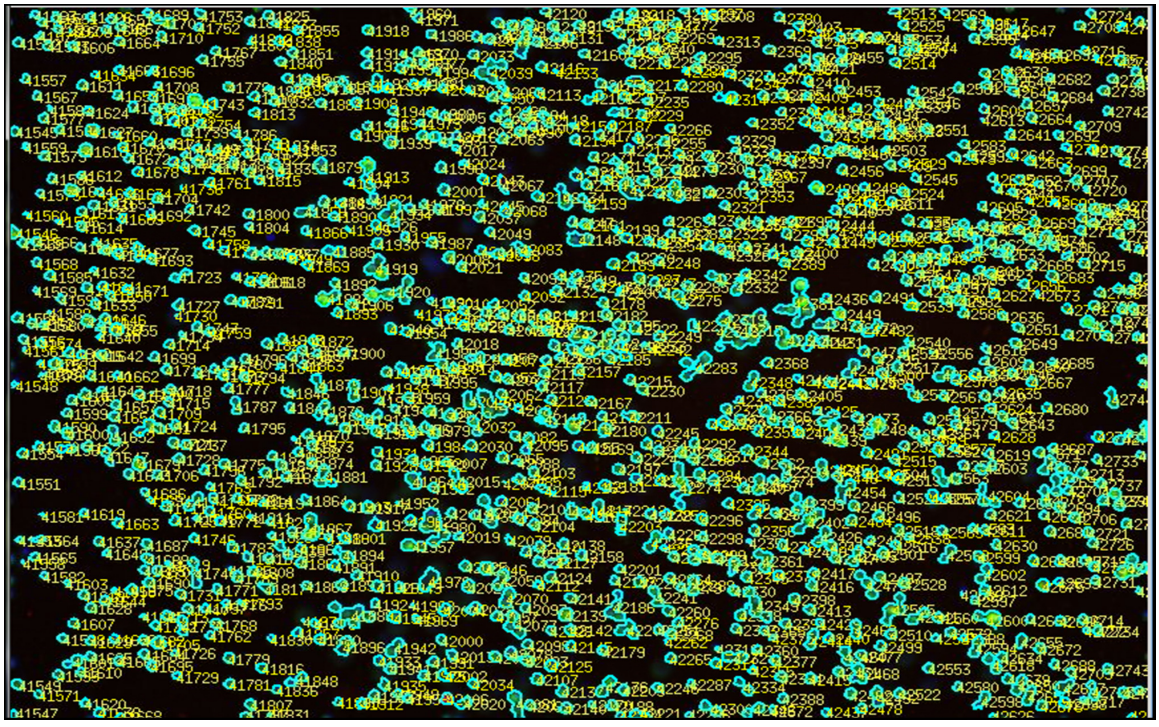


Figure 7: Edge detection shown segmenting and enumerating cells on an entire panel.

Consider the center pixel in a 9×9 grid. The connection region can be set to being

any of the eight pixels around or the four pixels side-by-side with the pixel at hand. With this information an outer edge is formed around an entire cell by seeing a closed loop of connected pixels. This elementary approach works well for the our data.

When determining pixels' connectedness, one may use a 4-pixel or 8-pixel connection parameter. The 4-pixel parameter deems pixels connected if they lay side-by-side, sharing a direct border. With the 8-pixel parameter, pixels are connected if they lay side-to-side or diagonal, with corner touching corner. We chose to use the 4-pixel connection parameter because we reasoned that there would be less classification of multiple cells as a single event.

The other thing to note is that one needs to be careful not to resize the initial pictures. There are fundamentally only 250×250 data points (pixels) of real data. Any resizing would involve some interpolation which brings with it an estimation of the pixel value between two points. It is with this consideration that we do all of our calculations from a pixel-centric point of view. We take caution not to add any unnecessary bias. Once the cell boundaries (or cropping noise boundaries) are detected, then we have completed our first step in classifying an event. The next step is to determine a feature descriptor for all pixels inside the event. A feature descriptor, as we have seen, is a vector describing some local curvature and intensity information about a specific pixel.

The code to develop the images with event detection boundaries and enumeration is given in the appendix Section A.1.

5.2 CTC and WBC feature descriptors

Section 3.2, we mentioned that one of the desirable traits of a feature descriptor is repeatability, meaning that a descriptor which could consistently identify an element under rotation and lighting changes.

With this in mind, we took on the task of building unique feature descriptors from the ground up. The main benefits of this are to increase our understanding of the data and descriptors, to build comprehensive pixel-centric descriptors and to have the control to increase the algorithm's efficiency and storage burdens.

5.2.1 Custom construction of feature descriptors

Suppose for a given image layer, we would like to construct a feature descriptor for a given pixel, p^*. Let $\Theta \in \Re^{9\times 9}$ be a nine-by-nine matrix of gradient angles whose center pixel is p^*. Let $\boldsymbol{M} \in \Re^{9\times 9}$ be a matrix of the associated gradient magnitudes.

The code developed to compute both of these quickly using a convolution technique is given in appendix Section A.2.

It is at this point our construction of a feature descriptor begins to diverge from the standard literature. We compute a feature descriptor for every pixel in the region of interest. For this task, odd-numbered grid size (e.g. 9×9) is necessary because we want pixel-centered feature descriptors. The image below shows the contrast in our feature descriptors grid compared with that seen in Lowe [20].

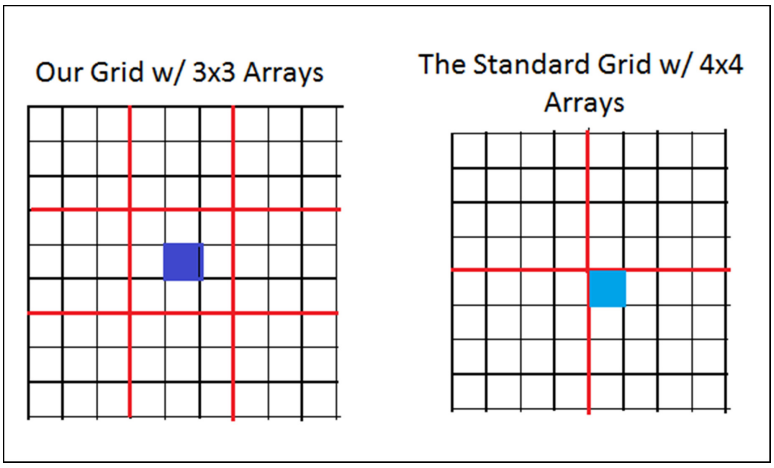


Figure 8: The blue pixel in the grid represents the pixel we are going to describe. In the literature 16×16 grids are more commonly used.

In the most general sense, the feature descriptor is some function specified for p^* that maps gradient information (direction and magnitude) to a $k \times 1$ vector. That is

$$f(p^*, \vec{\boldsymbol{M}}, \Theta) \in \Re^{k\times 1}.$$

The specifics of how we constructed the feature descriptors is explained below. We

introduce some notation below in hopes to bring clarity.

First, we subdivide our angle and magnitude matrices into 3×3 sub-matrices. Let $\{A^{(\Theta)}, B^{(\Theta)}, \ldots, I^{(\Theta)}\}$ be the 3×3 sub-matrices forming Θ as such

$$\Theta = \begin{pmatrix} A^{(\Theta)} & B^{(\Theta)} & C^{(\Theta)} \\ D^{(\Theta)} & E^{(\Theta)} & F^{(\Theta)} \\ G^{(\Theta)} & H^{(\Theta)} & I^{(\Theta)} \end{pmatrix}.$$

Similarly, let $\{A^{(\boldsymbol{M})}, B^{(\boldsymbol{M})}, \ldots, I^{(\boldsymbol{M})}\}$ be the 3×3 sub-matrices forming $\boldsymbol{M}$ as such

$$\boldsymbol{M} = \begin{pmatrix} A^{(\boldsymbol{M})} & B^{(\boldsymbol{M})} & C^{(\boldsymbol{M})} \\ D^{(\boldsymbol{M})} & E^{(\boldsymbol{M})} & F^{(\boldsymbol{M})} \\ G^{(\boldsymbol{M})} & H^{(\boldsymbol{M})} & I^{(\boldsymbol{M})} \end{pmatrix}.$$

For each of the nine direction sub-matrices, $\{A^{(\Theta)}, B^{(\Theta)}, \ldots, I^{(\Theta)}\}$, we build a 4×1 aggregate gradient magnitude vector. We chose them to be of size 4×1 because the elements will contain magnitude information related to one of the four natural cardinalities: east, north, west, south. These vectors will hold the relative gradient information with respect to the central pixel, p^*. To make this more clear, let us assign these vectors names. For the submatrix $A^{(\Theta)}$ and $A^{(\boldsymbol{M})}$, we form the 4×1 vector $\vec{A}$. For the submatrix $B^{(\Theta)}$ and $B^{(\boldsymbol{M})}$, we form the 4×1 vector $\vec{B}$. We continue this naming convention for the remaining sub-matrices.

The second step involves developing a method with the rotationally invariant property. This means we need to incorporate the gradient direction information of p^* into the construction of all the components in p^*'s feature descriptor. We base this off of the fact that the descriptor is formed specifically for this pixel.

To clarify things in our notation, the gradient direction of the center pixel, p^* is

$$\Theta_{p^*} = E^{(\Theta)}_{2,2}.$$

We consider the major cardinal direction of Θ_{p^*}.

$$\Theta_{p^*} \in \begin{cases} [\frac{-\pi}{4}, \frac{\pi}{4}) & \rightarrow \text{East Pointing} \\ [\frac{\pi}{4}, \frac{3\pi}{4}) & \rightarrow \text{North Pointing} \\ [\frac{3\pi}{4}, \frac{5\pi}{4}) & \rightarrow \text{West Pointing} \\ [\frac{5\pi}{4}, \frac{7\pi}{4}) & \rightarrow \text{South Pointing} \end{cases}$$

This image may aid in describing the direction choice.

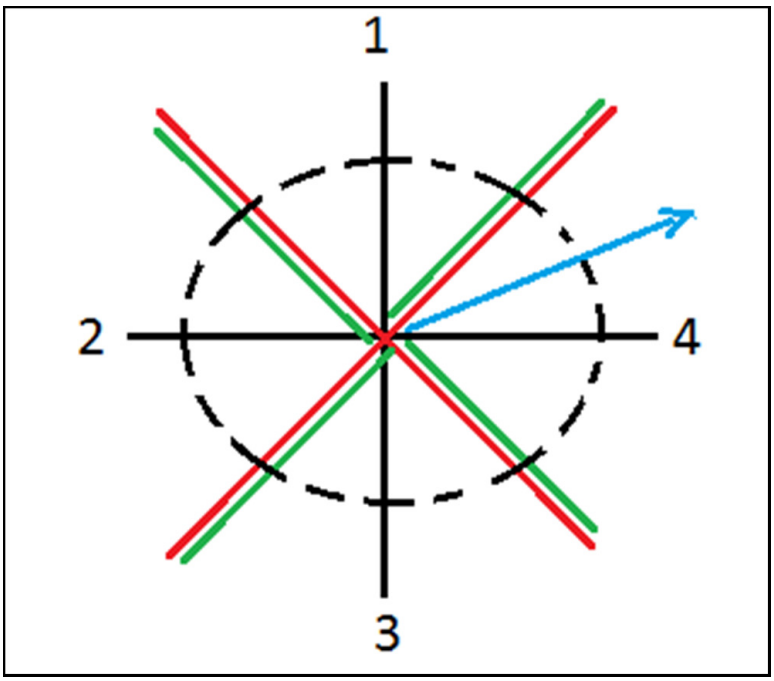


Figure 9: The pixel gradient represented by the blue arrow is in the range of $[\frac{-\pi}{4}, \frac{\pi}{4})$ and therefore gets assigned to East.

We will use the cardinal direction of the center pixel to build a correspondence between the element positions in the 4×1 vectors and the cardinal directions. Namely, the first element in the aggregate vectors $\vec{A}, \vec{B}, \ldots, \vec{I}$ will correspond to the cardinal direction of Θ_{p^*}. The second element in the aggregate vectors will correspond to the next cardinal direction (counter clockwise) from Θ_{p^*}. The third, the next cardinal direction from that, and finally the fourth element correspond to the last cardinal direction.

Now that we have some rotational information built into the element order of these vectors, our next step is to complete these vectors. For the sub-matrix $\boldsymbol{M}$, we aggregate the gradient magnitudes by summing the magnitudes of the gradients pointing in the same cardinal direction to form $\vec{A}$. And so forth for each sub-matrix.

For example, let $\Theta_{p^*} \in$ west, then

$$\vec{A} = \begin{pmatrix} \sum_{A_{ij}^{(\Theta)} \in \text{ west}} A_{ij}^{(\boldsymbol{M})} \\ \sum_{A_{ij}^{(\Theta)} \in \text{ south}} A_{ij}^{(\boldsymbol{M})} \\ \sum_{A_{ij}^{(\Theta)} \in \text{ east}} A_{ij}^{(\boldsymbol{M})} \\ \sum_{A_{ij}^{(\Theta)} \in \text{ north}} A_{ij}^{(\boldsymbol{M})} \end{pmatrix}$$

Finally, we concatenate all nine of our 4×1 aggregate magnitude vectors $\vec{A}, \vec{B}, \ldots, \vec{I}$ to get our 36×1 feature descriptor vector for the pixel p^*.

$$f(p^*, \vec{\boldsymbol{M}}, \Theta) = \begin{pmatrix} \vec{A} \\ \vec{B} \\ \vdots \\ \vec{I} \end{pmatrix}$$

We describe every pixel in a cell using the above method in each of the three images. We then stack the three 36×1 vectors to describe the pixel with a 108×1 feature descriptor. The final MATLAB code implemented to build these feature descriptors is shown in appendix Section A.3.

5.3 Building a cell feature descriptor dictionary

We build feature descriptors based on every pixel of 45,000 detected events. To compress this data, we perform K-means on these to define the key 'types' of feature descriptors (the cluster centroids). These key types will determine 'words' in a dictionary which will be necessary in building our term vectors. The MATLAB code written to accomplish this is

given in appendix Section A.4.

The number of words, N, is found using the elbow plot method with the Cosine metric as discussed in section 4.1.1. Figure 10 is the result of the distortion error with a red circle drawn around the value of N we chose to use.

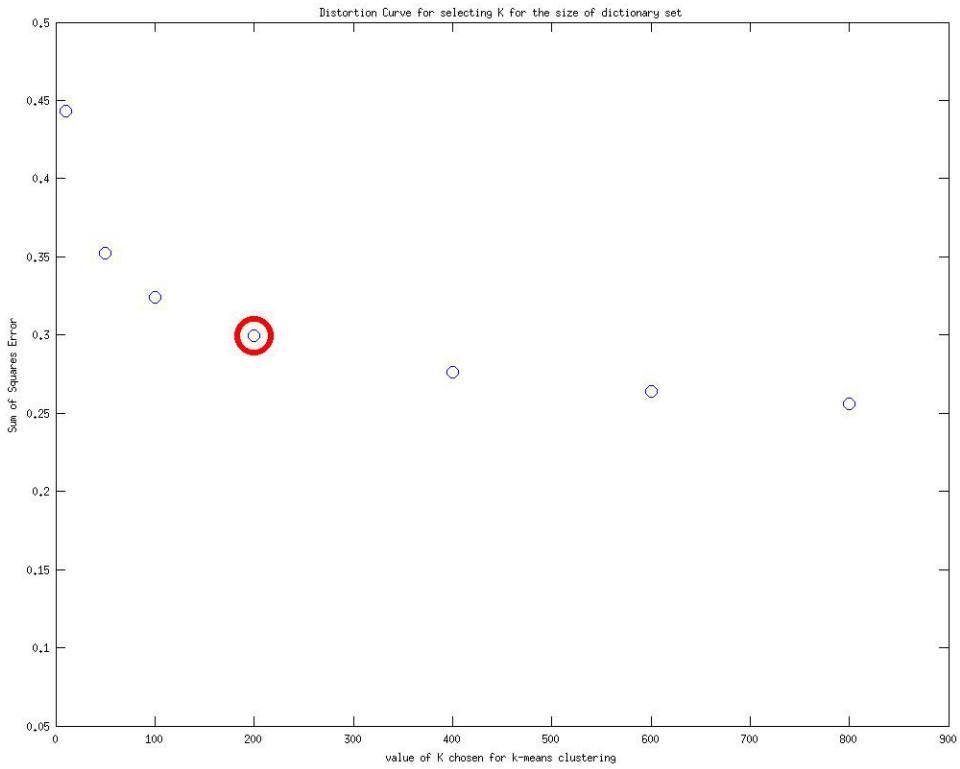


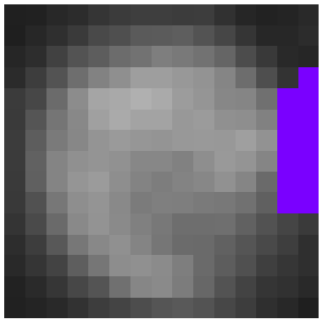
Figure 10: With the value of K plotted against the sum of squared errors (SSE), we see that at $K = 400$ the rate of reduction in SSE begins to diminish. We therefore choose our results to have 400 clusters, implying that our dictionary will be made up of 400 words (cluster centroids) describing types of feature descriptors (the clustered data points).

5.3.1 Pixel to feature vector calculations

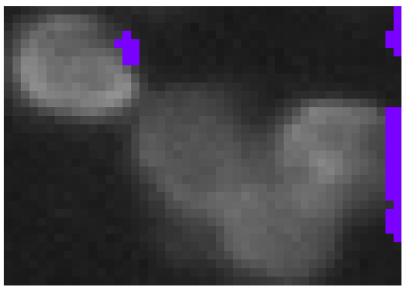
We now analyze these term vectors in more detail. The pixel colors are derived from which word is at the center of their feature descriptor's assigned cluster. For example, all turquoise colored pixels share similar feature descriptors and are clustered together. Figure 11 shows the pixels associated with a particular word from the feature descriptor dictionary. These

pixels are colored in purple. Figure 12 and Figure 13 show the pixels associated with other particular words in the feature descriptor dictionary.

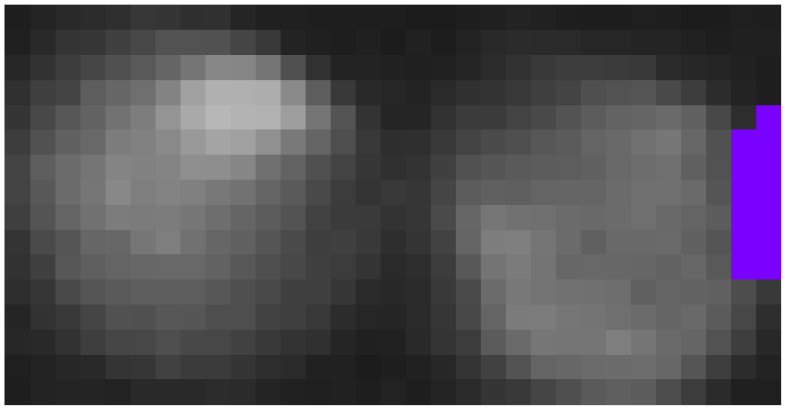
An exciting result is seen in these figures: the words in the dictionary are beginning to show meaningful interpretations. Related feature descriptors form in similar areas of cells. In Figure 12 and Figure 13, the related pixels all show in the right side of cells, even for the multiple cell event. The pixels show similarities with one another based on their location.



(a) Word 1 shown on WBC

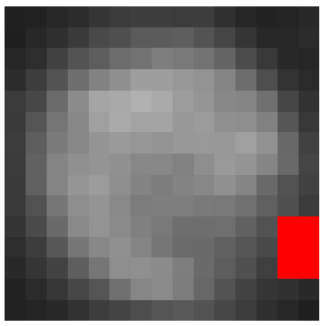


(b) Word 1 shown on a CTC event

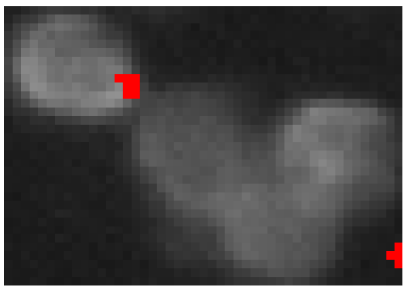


(c) Shown on multiple WBCs

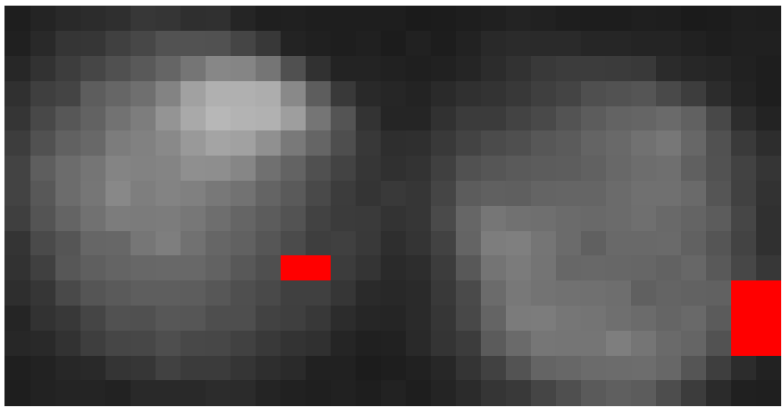
Figure 11: This figure shows the pixels of various events colored if their feature descriptors belong to the same clustering.



(a) word 2 shown on WBC

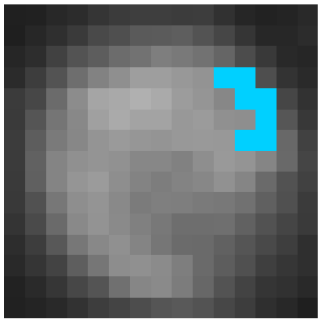


(b) Word 2 shown on a CTC event

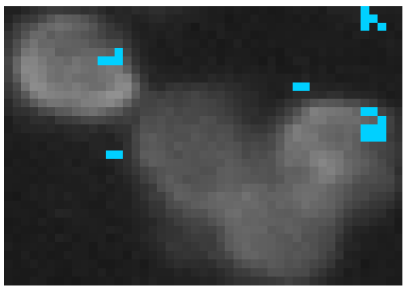


(c) Shown on multiple cells

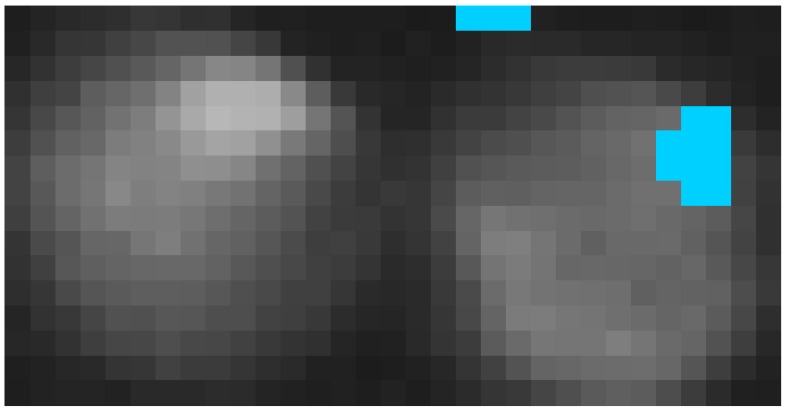
Figure 12: This figure shows the pixels of various events colored if their feature descriptors belong to the same clustering.



(a) word 3 shown on WBC



(b) Word 3 shown on a CTC event



(c) Shown on multiple cells

Figure 13: This figure shows the pixels of various events colored if their feature descriptors belong to the same clustering.

5.4 Assembly of term vectors

Now that we have a dictionary of words, our next step is to build term vectors for each new event. We proceed to the specifics of the method discussed in Section 3.3.

This is simply a matter of relating each pixel's feature descriptor to its associated word, then placing the counts of each word's occurrence in the corresponding term vector. The image below clarifies this concept.

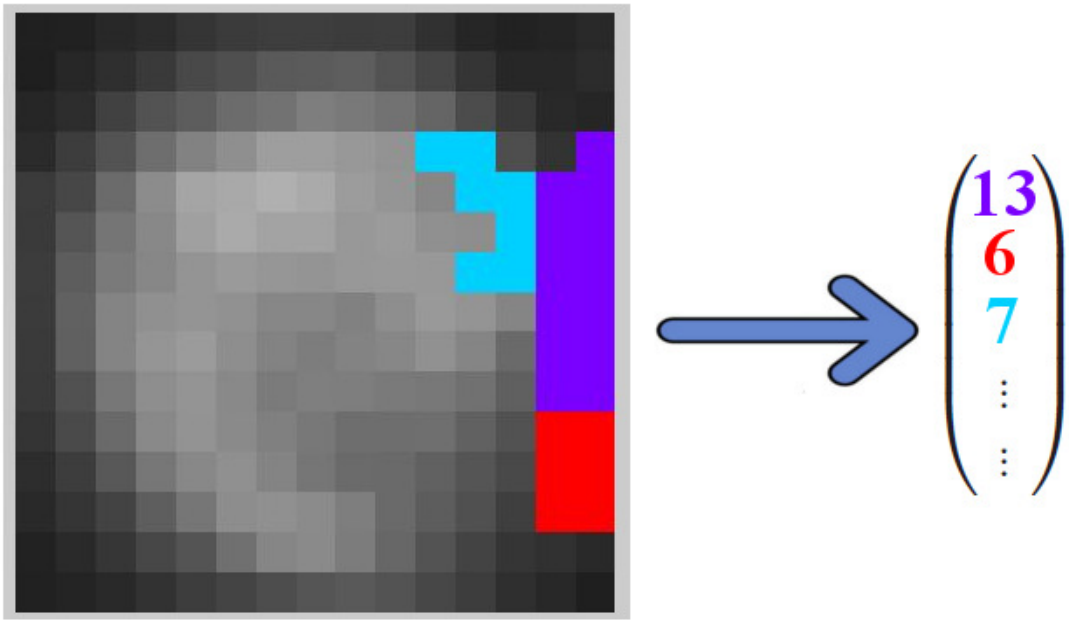


Figure 14: The event is shown on the left and the building of its associated term vector on the right. Each pixel is colored based on which word best represents it. Then the total number of word occurrences is expressed in a count vector called the associated term vector. As exampled above, the 13 purple colored pixels share similar feature descriptors and are clustered together. The center of that cluster is the word and corresponds to a specific element in the term vector. This element will have an entry of 13, for the 13 occurrences of this feature descriptor word.

The algorithm performs this task for every event detected. To see the full code of this procedure, please see appendix Section A.5.

5.5 Cosine rather than Euclidean metric

We now consider why we choose to build term vectors from words based on a K-means clustering with the Cosine metric rather than the Euclidean metric. The figures below show the classification of pixels into the most commonly seen word for each metric type. That is,

for each metric type, we color all the pixels whose feature descriptors were classified into the largest cluster found by the K-means algorithm.

(a) K-means results with cosine metric (b) K-means results with Euclidean metric

Figure 15: These figures show the difference in the two distance metrics for clustering feature descriptors. The amount of pixels belonging to the largest feature descriptor cluster are colored. Please note that the euclidean distance does not allow the K-means algorithm to detect the sensitivity needed to show differences in the feature descriptors.

The resulting difference between the two metrics is significant. When using the Euclidean metric, most of the features are clustered into the same group. Detecting the diversity of feature descriptors allows for our method to begin understanding the distinct features of a CTC over a WBC. In clustering textual documents, the Euclidean is often inappropriate [2][13] and proves to be here as well. The Cosine K-means utilizes the sparsity of the feature descriptors and term vectors, making it highly efficient [13].

5.6 Determining subclassifications: prebuckets and buckets

Ideally, we would like to analyze the structure of all of the term vectors using the choice method of hierarchical clustering. However, hierarchical clustering's weakness lies in its expensive computational procedures. With a sample set of 5000 cells, and a real set of nearly 100,000 cells, this method alone is too inefficient. Therefore, we will first compress all the term vector information with K-means, then perform hierarchical clustering on the resulting term vector cluster centroids.

5.6.1 The use of K-means to determine prebuckets

Once again, we use the method of examining the distortion error curve to set the number of clusters. We find that for our test set of 5000 term vectors, there exist 30 appropriate clusters that will best describe and compress this data.

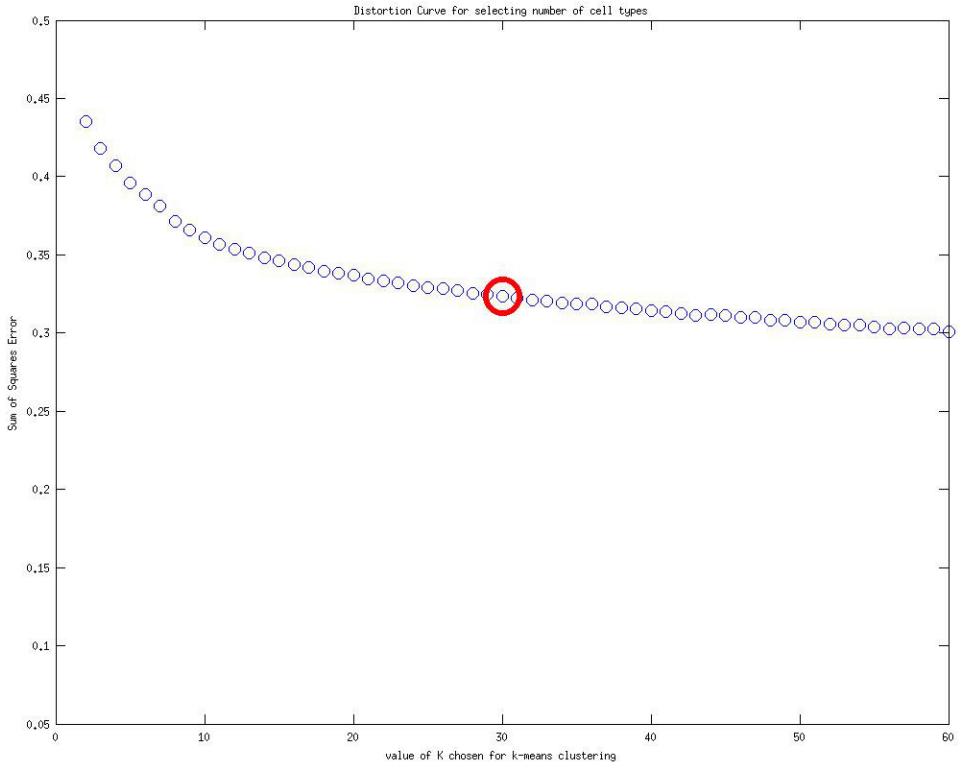


Figure 16: The elbow plot shown above helps us to identify the number of key types of term vectors. As discussed in Section 4.1.1 we take $K \approx 30$ to be the number of choice. These 30 clusters of term vectors are deemed the 'prebuckets' of our cell classification.

5.6.2 The use of hierarchical clustering to determine buckets

Now that our term vector data is compressed to 30 prebuckets, we perform hierarchical clustering to determine the relative similarities and structure. We find it best to present the relationship between prebuckets with a dendrogram. The details of dendrograms and their interpretations can be found in Section 4.2. The code for this analysis can be found in

appendix Section A.6.

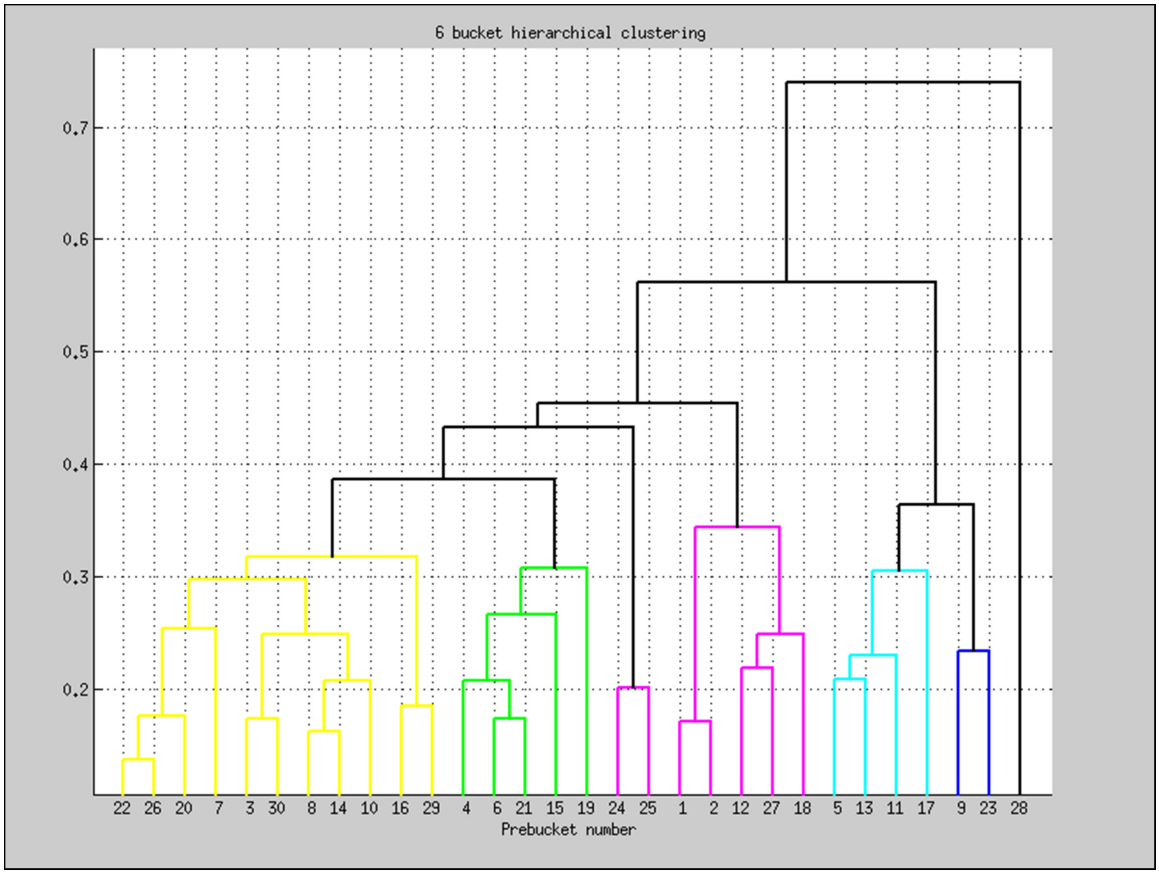
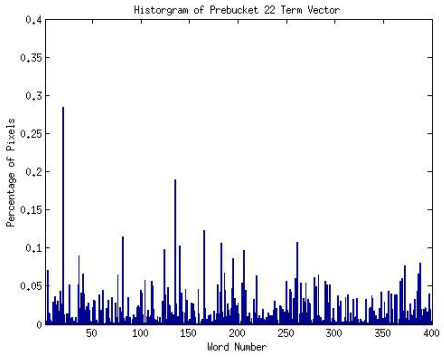
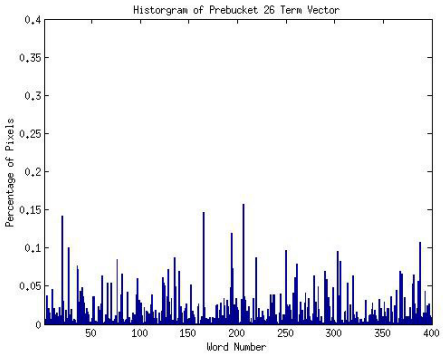


Figure 17: This dendrogram shows the relative similarities in term vector prebuckets. The numbers on the x-axis show the numeric labels of each of the 30 prebuckets. The heights of the path linking each prebucket show the relative cosine distance amongst the prebuckets. The tree was cut at a value of .35 giving the result of constructing 6 key 'buckets'. For example, prebuckets 4, 6, 21, 15, and 19 are joined together based on their relative similarity to form the green bucket. It is interesting to note that prebucket 28 is the most distant from all of the others. It is in fact so distant that it forms its own bucket.

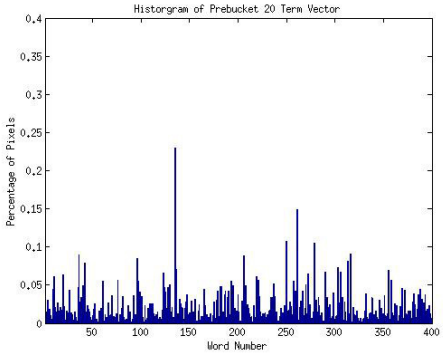
With each prebucket comes a prebucket centroid. This centroid helps to identify the shape of the term vectors that are most related to it. We present the centroid term vectors for prebuckets classified as similar in the dendrogram compared with the anomaly prebucket, respectively prebuckets 22, 26, and 20 (yellow in dendrogram, blue below) compared with prebucket 28 (black in dendrogram, red below).



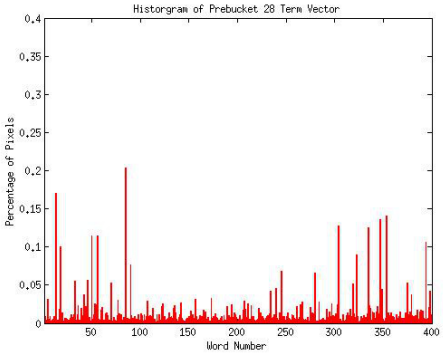
(a) Term Vector Centroid for Prebucket 22, shown in the yellow group.



(b) Term Vector Centroid for Prebucket 26, shown in the yellow group.



(c) Term Vector Centroid for Prebucket 20, shown in the yellow group.



(d) Term Vector Centroid for Prebucket 28, the anomaly prebucket shown in the dendrogram as the all black line.

Figure 18: Examples of prebucket term vector centroids for a common bucket (colored yellow in the dendrogram) and anomaly bucket 28 (black in the dendrogram).

Trying to interpret these term vectors with the naked eye may prove fruitless. However, in the next section, we consider which types of events are assigned to each of these buckets which may provide clear insight and interpretability.

6 Results and discussion

In Section 2.5, we introduced three main benchmarks that would help us determine the reliability and successfulness of any method built to classify CTCs. They were descriptive classification tests, non-trivial event detection, and rotational robustness. We will describe in this section our development of hypotheses, methods and results pertaining to these various tests.

6.1 Sensitive and specific classification

Our first benchmark, the capability to distinguish between event types, reflects the main goal of this research. We stated the test before as follows:

"When a given panel is input into the algorithm, the output should be groupings of similar events into various "buckets". These buckets should have some degree of interpretability and must at some level differentiate between white blood cells, circulating tumor cells and noise."

To develop a way to test this, we assessed what could be deemed successful. Two properties of a successful result would be

- sensitivity: there are buckets for the various types of events, and
- specificity: the majority of events in a bucket belonged to a certain cell type.

We've developed a method to build these buckets by considering natural clusters that exists in the space of term vectors. Hierarchical clustering allowed us to view this space's structure in the form of a dendrogram. From here, we 'cut' the branches of the dendrogram at an appropriate level which forms six buckets.

In order to classify the events in the buckets, we indexed each event and its respective type from the original image data. There are three main event types, but because some events contained multiple cells, we distinguished them by adding two additional types. Out of a total of 45,066 events, the classes are

- White Blood Cell (WBC) (count: 44,914)
- Circulating Tumor Cell (CTC) (25)
- Multiple cell detection: WBC attached to a CTC (14)
- Red Noise (54)
- Multiple event detection: WBC attached with red noise (58)

The following results show the first property, sensitivity, by showing the classification of events types from the full-sized panels.

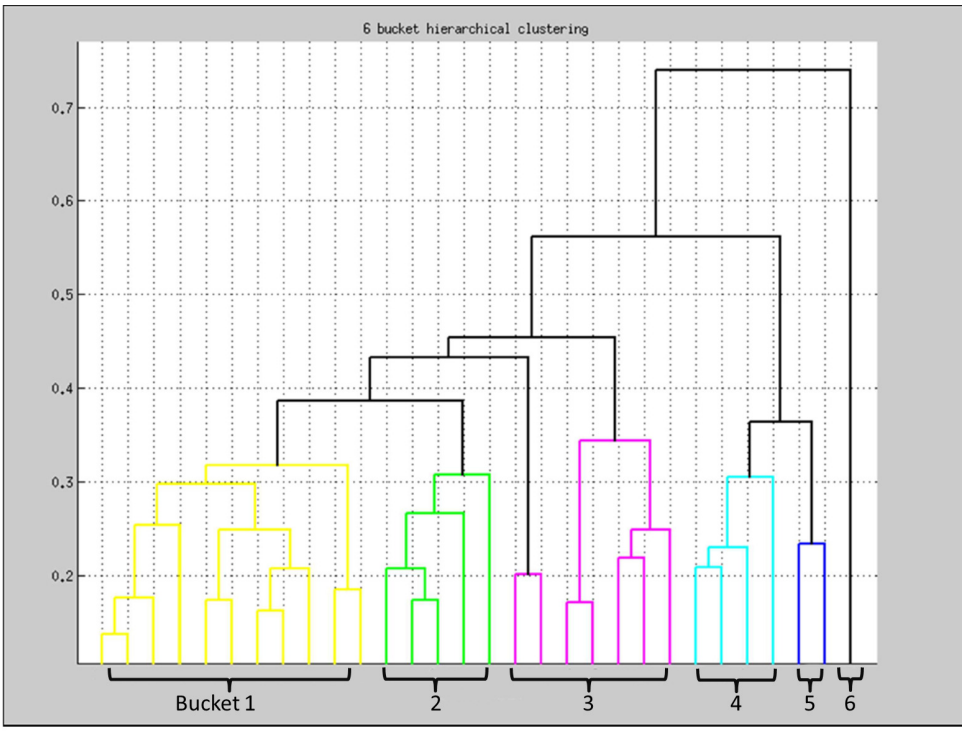


Figure 19: In this dendrogram, we label the 30 prebuckets by their bucket number. This will help us interpret the proceeding graphs.

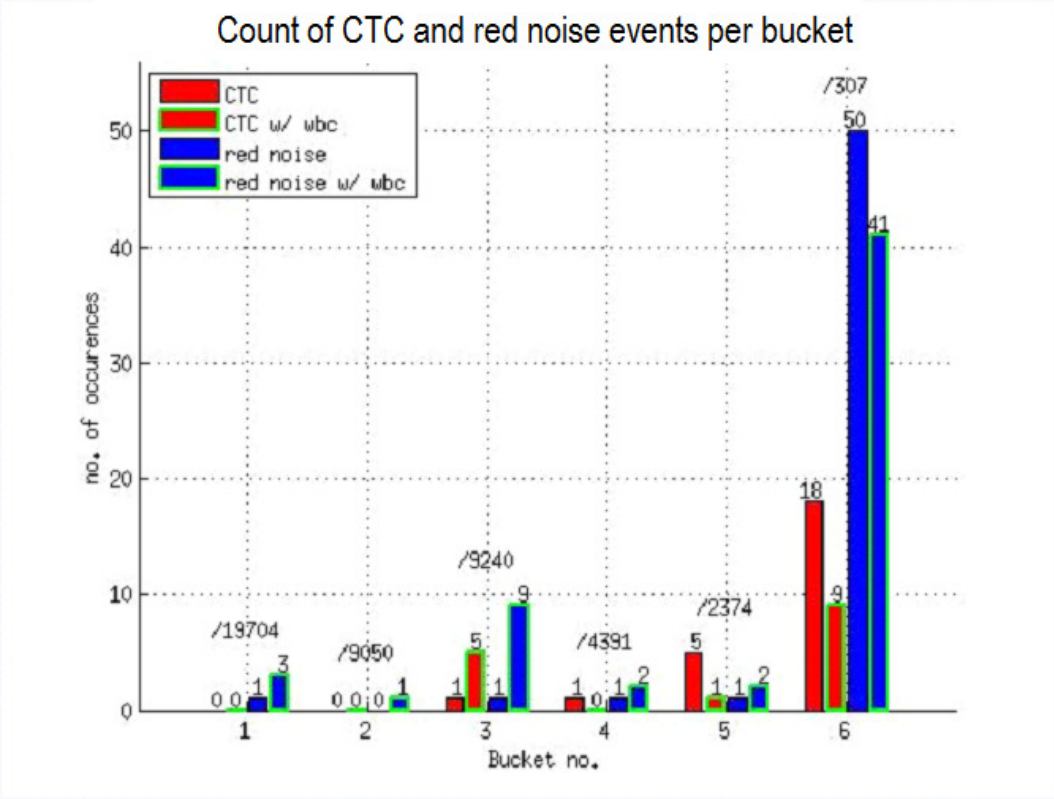


Figure 20: The count for CTC and red noise events are given for each bucket. From left to right, each colored grouping in the dendrogram corresponds to a bucket number shown here, from left to right. The yellow cluster of prebuckets on the far left side of the dendrogram are defined as bucket one, and so on. The anomaly black prebucket 28 corresponds to bucket six. Each bucket consists of all the term vectors assigned to the prebuckets assigned to it. The total number of events in each bucket is shown on top of each bar. (e.g. /19704, /9050, etc.)

The CTC classification is as follows

- 67.5 percent classification of events on CTCs into Bucket 6 (anomaly bucket)
- 32.5 percent classification of CTCs into other buckets.

It is key to note that a majority of the CTC and red event information is contained in the least populated bucket, bucket six. It is also important to recall the fact that bucket six was formed from the anomaly prebucket 28, the prebucket furthest away from all other 29 prebuckets. This means that the term vector space has an incredible amount of structure

that allows us to distinguish these two event types from the majority event type, the white blood cells.

The next figure helps showcase our method's specificity by showing the proportion of CTCs to other cells in each bucket.

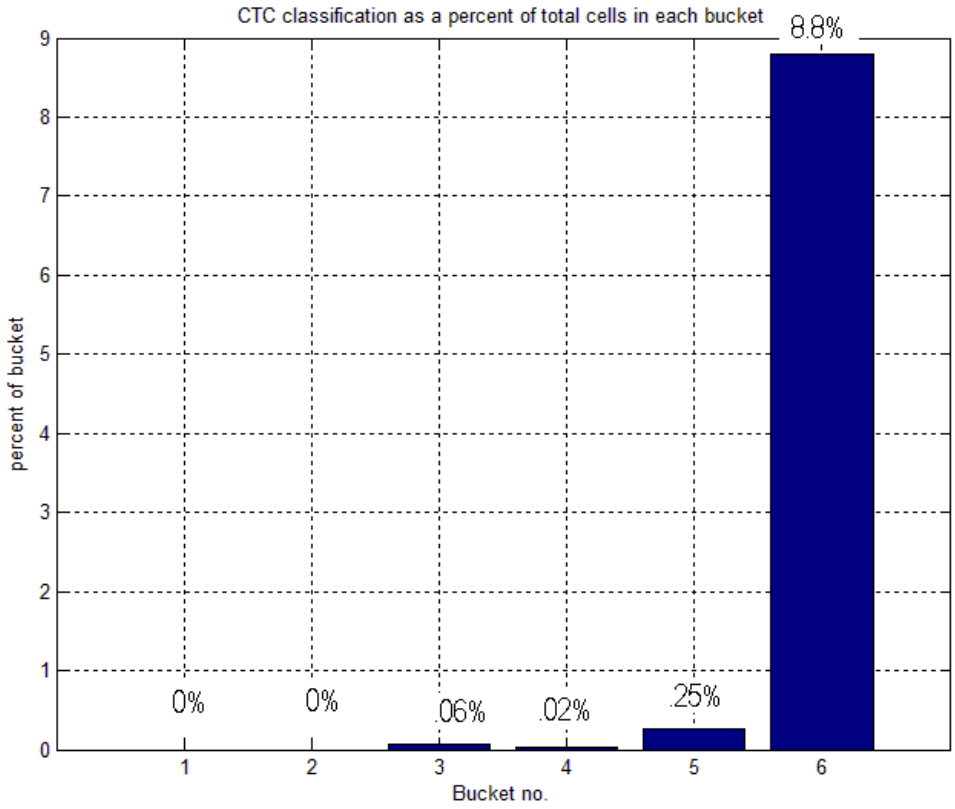


Figure 21: The CTC percentage within each bucket demonstrates that bucket six is highly distinguished by CTCs. Another way of saying this is that bucket six is highly specific to CTC information.

These two results together highlight the fact that bucket six, the anomaly and least populated bucket, captures the majority of CTC information and is heavily distinguished by CTCs. It is not only sensitive to capturing CTCs, but is specific in capturing CTCs and few other events. Ideally, we strive for capturing all CTC information in an exclusively CTC bucket, so there is still work to be done on this algorithm.

In the next result section, we will look into whether or not this structuring distinguished

events only on the basis of the presence of the red stain.

6.2 Nontrivial event detection

This section addresses the benchmark of non-trivial event classification. The classification results displayed above raised excitement for the feasibility of computer vision but the fact that CTC and red noise classification were so similar also raised skepticism. After all, the presence of Cytokeratin information or red coloring was a key feature in distinguishing CTCs. To be clear, we'd like to know

"Is this an overly complicated red color detector?"

We address this by considering the question of what happens when we leave out the information from the Cytokeratin image.

Our hypothesis is:

- If we leave the red information out from the feature descriptors and our CTC events cease to distinguish bucket six, then we are accomplishing nothing more than detecting the color red.

- However, if we leave the red information out from the feature descriptors and bucket six remains heavily comprised of CTC information, then our method does in fact accomplish more than red stain detection.

In conducting this experiment, we focused on the data CTC data from the cropped images. These are images formed from a 40x zoomed-in area around a known CTC. The data is as follows:

- There are approximately 4,900 WBCs.

- There are exactly 99 CTCs

Here are the results showing the classification consistency of CTCs with and without the red layer information anywhere in the algorithm's construction.

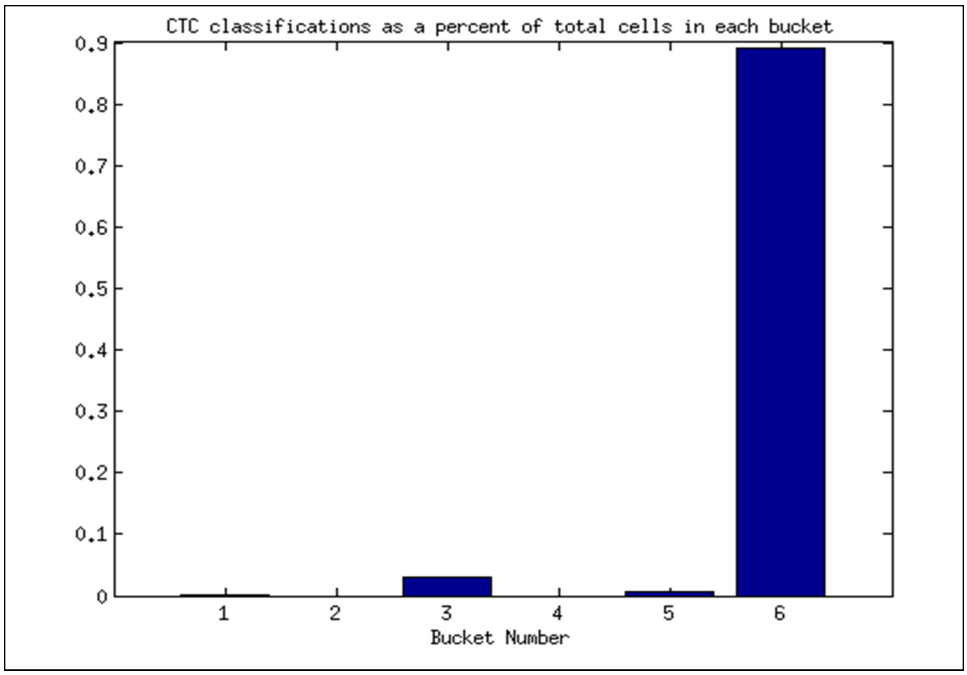


Figure 22: This figure shows the classification of CTCs as a percent in each bucket. The information is from cropped images enlarged around CTCs which contain very little red noise. Again, the anomaly bucket six is highly specific to CTC information.

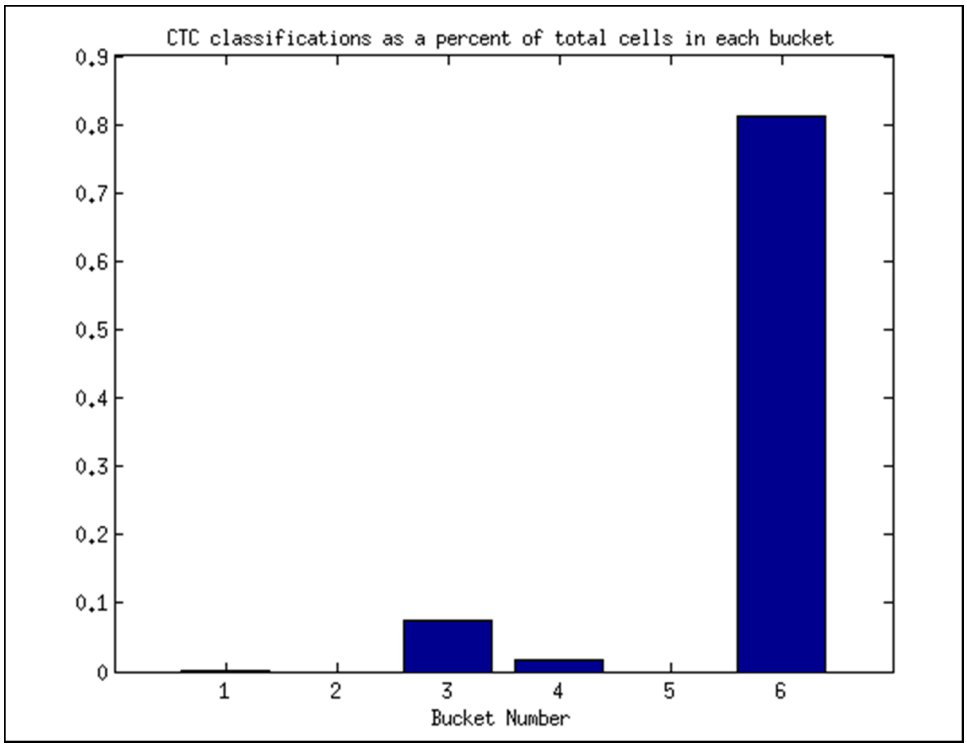


Figure 23: Classification as a percent in each bucket without the use of the Cytokeratin layer in the formation of the feature descriptors. This figure shows a comparable result to the figure above, displaying the proportion of CTCs to other cells in each bucket. However, these results are found without any Cytokeratin information used in the formation of feature descriptors, term vectors, or clustering.

The near-identical results satisfy the hypothesis given above and demonstrate that our algorithm accomplishes more than just classifying mainly with respect to the presence of red coloring. Unfortunately, these results do not inform us why red noise clusters so similar to CTC information. But it does inform us of another interesting classification conjecture: There exists a sufficient difference between CTCs and WBCs within the DAPI and CD-45 channels to get good classification distinction.

In the next section, we turn to the final benchmark of our algorithm, rotational robustness.

6.3 Rotational invariance

The orientation of a cell on a slide should, ideally, not impair its characterization in terms of features. Because of this fact, we deem it necessary to develop a method which would maintain consistency in cell classification under rotation. That is, any cell should be classified into the same bucket before and after any rotation. We kept this in mind as we built the feature descriptors. But up until now the effectiveness of our feature descriptor design has not had to undergo scrutiny of a formal test.

To test for rotational invariance, we indexed 100 cells, both CTCs and WBCs, from three cropped panels. After running the algorithm and keeping track of each cell's assigned prebucket (out of a total possible of 30 prebuckets), we rotated the panels 90 degrees counterclockwise then ran the algorithm a second time. We kept track of these new prebucket assignments for each cell. We repeated this process two more times, rotating the panels at 180 and 270 degrees.

Our hypothesis:

- If a sufficiently large proportion of cells are assigned to the same prebucket across the four rotation angles, then our algorithm displays invariance to panel rotation. One can make the case that the algorithm applied to cells themselves may still not display complete rotation invariance. For example, if an individual cell showed inconsistent classification under a rotation of just 15 degrees, then this test of 90 degree rotations would not suffice to catch that. We therefore clarify the that we are at this point seeking rotation invariance at the panel level, and will consider cell rotation under various angles once this benchmark is met.

The results of a comparison in prebucket classification consistency are shown below.

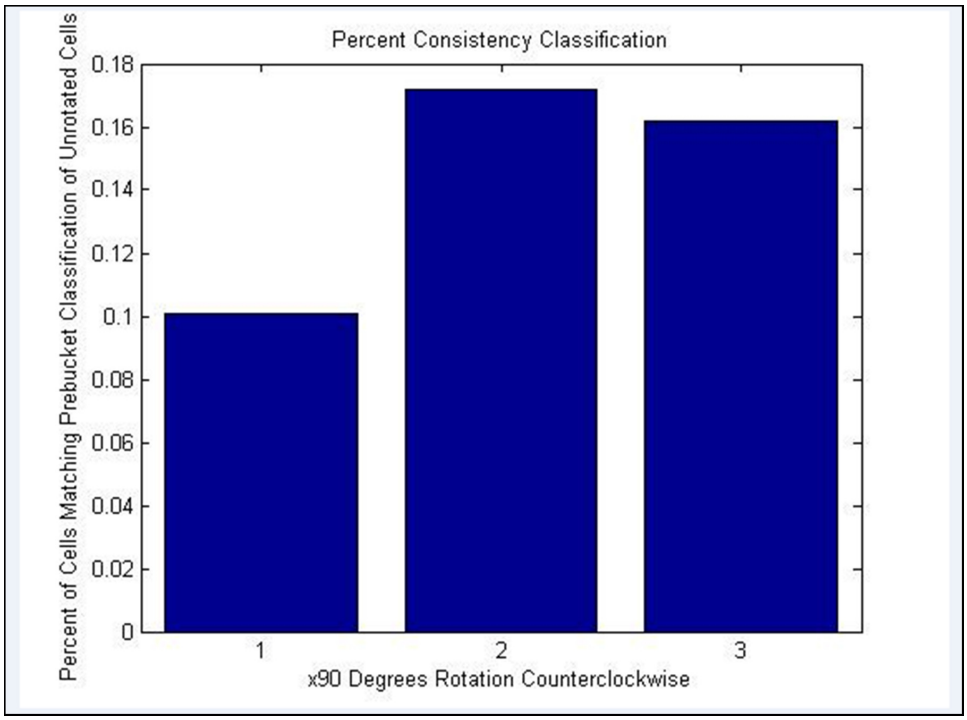


Figure 24: The percent of events consistently classified to their original prebucket is shown for each of three rotations (90, 180 and 270 degrees). As one can see, the shockingly low classification consistency implies that we were unsuccessful in building rotational invariance into our object classification when we created the feature descriptors in Section 5.2.

In looking at 99 cells that have been rotated 90, 180, and 270 degrees counter-clockwise we never obtain more than 20 percent consistency with the prebucket classification of the cell in its original orientation. This percentage is small but still yields better results than if the cells are randomly assigned. (probability being $1/30 = .0333 < .1$.) This result shows that our algorithm in its current state does not exhibit the desired property of panel rotation invariance. We therefore note that our algorithm is in need of further development. In the upcoming modification section, Section 7.4.1, we reference the current work being done on this improvement.

7 Examination of general method

It is at this point in the thesis that we address the method in general. In this section, we examine the strengths, weaknesses, added value, and future modifications of the processes presented in the paper.

7.1 Strengths

7.1.1 Use of a data-driven methodology

Systematic studies [35][36] classify detection methods into the three broad categories of qualitative model-based methods, quantitative model-based methods, and process history-based methods.

First and easiest to understand, qualitative model-based methods help visualize system relationships through flowcharts, fault trees, digraphs, etc. The main objective of this method is to visualize in a broad sense the loosely defined relationships among a system's elements. Secondly, a quantitative model-based method predicts anomalies by reasoning through some fundamental understanding of processes in the system. Model developers use their understanding of the physics of a system to formulate large sets of mathematical equations expressing all known relationships. In contrast to these first methods, a process history-based method assumes only what is seen in large amounts of historical data from a system. By the use of pattern analysis, classification, and prediction algorithms, this methods links new real-world observations to trends seen in the past.

A quantitative model-based approach is ill-fit for our system for several reasons. Mainly, we do not have the fundamental mathematical relationship equations to differentiate cells. Even if all the physical relationships were known, a mathematical model still lacks a complete understanding of unstructured uncertainties in the image and staining process itself. The lack of a fundamental morphological structure has caused the many differing descriptions by operators and would severely impact any model-based method.

In contrast, the large amounts of historic image data collected by the research give the

necessary fuel to run a process history based method. We formulate the CTC detection problem as a pattern-recognition problem on the space of cells and seek to extract structural information from this space through clustering methods. This is a key strength of the approach presented here. With only raw image data, this data-driven method can extracted information to distinguish and classify cells. The method was built completely from real data, so there was no hypothetical morphological model built that would cause us to worry about oversimplification.

7.1.2 Pixel-centric

In an attempt to incorporate nearly all pertinent information in a raw image into our detection method, we designed the feature descriptors from the ground up to describe every pixel on an event. The computation power used to do this was immense, but seemed appropriate as no information would be excluded before the clustering analysis began.

7.1.3 Efficiency of classification

The method's two parts allow for most of the computationally expensive procedures to be performed once, then held offline and used to aid in a quick online part of classifying incoming events. The offline part consists of building a feature descriptor dictionary and clustering existing term vectors to gain an interpretation. The online part consists of reading in a new event and building the feature descriptors and term vectors then placing it into the appropriate prebucket. The clustering involved in the offline part is computationally expensive. It also requires a large amount of storage to perform in the case hierarchical clustering. The online part, however, has very few storage requirements. In fact, one could even describe an entire cell with the description as little as its associated $n \times 1$ term vector, though this does not give information on the location of each of these terms inside the cell. This term vector is nonetheless a description of the cell that compresses a large amount of information into a content-rich vector.

7.2 Weaknesses

7.2.1 Drawbacks to data-driven approach

A drawback to a data-driven approach is the fact that incompleteness in historic imagery makes it difficult to detect not yet seen CTC structures. A data-driven approach also lacks the ability to fully interpret why particular cell classifications occurred, being as it simply classifies by past trends instead of a deeper understanding of the system.

7.2.2 Rotational invariance

We saw above in the results Section 6.3 that our method's rotational invariance proved inadequate when fully examined. The modification Section 7.4.1 presents some of the research currently being done.

7.2.3 Higher dimensional insight lost with data compression

Although the term vectors are very content rich in that they house adequate information to determine bucket classification at our level of accuracy, they are a major compression of the data. Each cell was originally a 400 (20×20) pixel patch and we assigned it a 400 element term vector. One may think there was a compression rate of 1:1. However, it was the expression of each pixel in terms of a feature descriptor that allowed us to describe how parts of cells relate to other parts of cells. So in actuality, the compression rate must involve the creation and formation of the feature descriptors. Considering just the numbers behind what's going on, a cell image falls into roughly a 20×20 pixel grid. A feature descriptor of size 108×1 is ascribed to each of these pixels. This is a total of approximately 40,000 elements of information. So a compression of a cell's feature descriptor into a 400×1 term vector is a compression of nearly 100:1.

7.2.4 Parameters to tune

Though the method is data-driven, there are still parameters that should be fine-tuned for optimal results. The key parameters to tune are the number of words in the dictionary, n, and the effect of other feature descriptor designs. Though we presented the elbow method in Section 4.1.1 as a proposed way to determine the number of centers to preassign in the K-means algorithm, this value was only roughly optimized over a small number of n values in the case of determining the dictionary size. Further tuning of these parameters would be to use a testing set of images and find the values and design which produce the best classification results.

7.2.5 Speed of algorithm should be improved

To speed up the algorithm to clinical working standards, the online component of the algorithm needs improvements. The ordering of cells is not important when building feature descriptors and term vectors. This lends itself well to a parallelized algorithm.

7.3 Added value

7.3.1 Automated CTC detection

This method set forth a viable way to directly apply the field of computer vision to the study of blood cells as related to oncology. The key success of this first application, though the results are far from the necessary precision, is the fact that the process was entirely automated. From start to finish, this method loaded in the image, detected events, and ascribed on a pixel-by-pixel basis local curvature information sufficient enough to draw main distinctions between most CTCs and white blood cells. Once these distinctions were made, they were applied again to a set of new events, predicting with fair accuracy the type of each cell. This automation can play a first role in sorting through the millions of cells imaged by the pathologists at the Scripps Research Institute and other institutes. With saved time, their skills can be used to advance their field in new ways.

7.3.2 Sub-classification of tumor cells

Drawing distinctions between white blood cells and CTCs has advantages discussed throughout this thesis. Furthering these results can lead to information distinguishing possible types of CTCs. Attempts are already being made at exploring clusters of branches stemming from the anomaly branch 28. The framework has been laid to go about subclassifying CTCs with a rigorous methodology. This process, now that it has been fixed and results are being outputted, may be improved to deepen the exploration of CTC subclassification.

7.3.3 Determination of effects of various types of chemotherapy

When first working with the pathologists at Scripps, there was much excitement and exhortation to dig deeper into these CTC subclassifications. The main reason for this enthusiasm was a long-term dream the researchers had to build a methodology to rigorously distinguish CTCs affected by various types of chemotherapy radiation [17]. If they could precisely classify various types of CTCs, then they could see how the different radiation treatments affect the differing CTCs. This long-term goal has enormous impact on the way doctors could understand both cells and the treatment results.

7.4 Future modifications

Although the methodology in this thesis has achieved several of the goals its set out to reach, there are clear paths ahead for various modifications and improvements. We will cover four main ones that we have identified as next steps for improvements, describe the problems of the unmodified result and may, in some cases, describe the next steps being implemented.

7.4.1 Full rotational invariance

The rotational robustness designed into the construction of the feature descriptors as described in Section 3.2 does not suffice in consistently classifying cells under rotation. Out of the three goals we set out to reach, the results displayed in Section 6.3 show that this is the

only goal that was not met.

One may adopt new design techniques and change the way feature descriptors are built from nine-by-nine grids to nested circles. This is the subject of future work.

7.4.2 Subclassification of CTCs by identifying further tree breaks

As stated in the added value Section 7.3.2, a deeper understanding of the various types of CTCs would prove invaluable for many areas of oncology research, both in the pathological study and in treatments. An obvious next step for this improvement is to explore the dendrogram's branch 28. We would use the results from a further hierarchical clustering on all cells classified in that bucket. Again, the hierarchical clustering contains all the structuring information between the cells, so we predict that natural clusters within that bucket exists and will prove insightful. This may help to identify the discrepancies between CTCs and red noise events that are classified into prebucket 28.

7.4.3 Reducing missed detections by multiple passes

Given the nature of the data at hand, it is far more desirable to have an algorithm which errs on the side of more false positives than missed detections. That is, the presence of any cancer cell is critically important, so, ideally, we'd like to have 100 percent of the CTC cells classify into a specific prebucket and sort out the prebucket later. Therefore our results indicating that 13 CTCs were classified into buckets three, four and five are unfavorable. Our remedy for this was to consider the nearest three prebucket centers to each cell's term vector. If any of these were in prebucket 28, then we bin the cell into that prebucket. A quick implementation of the MATLAB code needed for performing this multiple pass check is given in appendix Section A.7. The success of the results are shown in the table below:

7.4.4 Improved edge detection

In test runs over thousands of cells, it came to our attention that the initial edge detection missed detecting some cells on images. The problem lay within the first step of the edge

Table 2: Results of multiple passes

Classification into Bucket Six			
No. of Passes	CTCs found	Total Events	Percent of All CTCS
1	27	307	67.5%
2	30	356	75%
3	32	<400	80%

detection, in turning the RGB panel information to an appropriate gray-scale image and then to a binary image. The problem resulted from the fact that the red and blue levels are mapped to a value closer to black than to white. When the image is then turned into a binary image, the thresholding value was unpredictable across data sets, which led to large portions of blue (DAPI) and red (Cytokeratin)image information being left out.

This issue was resolved by forgoing the RGB to black and white mapping in total, and instead considering the unioned regions between all three layers as separate inputs to the binary cut-offs. After all, the key idea is to locate the various CTCs which are distinguished mostly by the presence of these stains. This modification is being implemented, but those results will be presented elsewhere.

8 Concluding remarks

We close with the statement quoted at the beginning of the paper, emphasizing the overarching motivation behind our work.

'Detection and characterization of these cells is a promising method for both diagnosis and clinical management of cancer patients as well as monitoring treatment.'

In an attempt to aid in that work of detecting and characterizing, the computer vision methodology developed in this paper took aim to lay out a potential path for building an automated image detection technique where CTCs could be objectively distinguished and classified.

We hope that the novel approach presented in this paper will help guide researchers and clinicians in their pursuit of understanding CTCs.

References

[1] 21st International Conference on Patter Recognition, *Contest on hep-2 cells classification*, http://www.http://mivia.unisa.it/hep2contest/index.shtml, 2012.

[2] J. Ghosh A. Strehl and R. Mooney, *Impact of similarity measures on web-page clustering.*, AAAI 2000 Workshop on AI for Web Search (2000).

[3] LWMM Terstappen AGJ Tibbe, MC Miller, *Statistical considerations for enumeration of circulating tumor cells*, Cytometry Part A 71A: (2007).

[4] E. Cordelli and P. Soda, *Color to grayscale staining pattern representation in iif*, Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on, June, pp. 1–6.

[5] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, *Visual categorization with bags of keypoints*, Workshop on statistical learning in computer vision, ECCV, vol. 1, 2004, p. 22.

[6] C. Ding and X. He, *K-means clustering via principal component analysis*, Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 29.

[7] P. Elbischger, S. Geerts, K. Sander, G. Ziervogel-Lukas, and P. Sinah, *Algorithmic framework for hep-2 fluorescence pattern classification to aid auto-immune diseases diagnosis*, Biomedical Imaging: From Nano to Macro, 2009. ISBI '09. IEEE International Symposium on, 28 2009-July 1, pp. 562–565.

[8] G. Attard et al. F. A. W. Coumans, C. J. M. Doggen, *All circulating epcam1ck1cd452 objects predict overall survival in castration-resistant prostate cancer*, Ann of Oncology 21: 1851 (2010).

[9] David A. Forsyth and Jean Ponce, *Computer vision, a modern approach*, Prentice Hall, 2003.

[10] Cyril Goutte, Lars Kai Hansen, Matthew G Liptrot, and Egill Rostrup, *Feature-space clustering for fmri meta-analysis*, Human Brain Mapping 13 (3): 165-183 (2001).

[11] Rico Hiemann, Thomas Bttner, Thorsten Krieger, Dirk Roggenbuck, Ulrich Sack, and Karsten Conrad, *Challenges of automated screening and differentiation of non-organ specific autoantibodies on hep-2 cells*, Autoimmunity Reviews **9** (2009), no. 1, 17 – 22.

[12] Tsu-Yi Hsieh, Yi-Chu Huang, Chia-Wei Chung, and Yu-Len Huang, *Hep-2 cell classification in indirect immunofluorescence images*, Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on, Dec., pp. 1–4.

[13] Yuqiang Guan Inderjit S. Dhillon and J. Kogan, *Refining clusters in high dimensional text data*, http://www.cs.gsu.edu/~wkim/index_files/papers/refinehd.pdf, 2002.

[14] Y.G. Jiang, C.W. Ngo, and J. Yang, *Towards optimal bag-of-features for object categorization and semantic video retrieval*, Proceedings of the 6th ACM international conference on Image and video retrieval, ACM, 2007, pp. 494–501.

[15] S.C. Johnson, *"hierarchical clustering schemes"*, Psychometrika 2:241-254 (1967).

[16] Peter Kuhn, *The kuhn laboratory: Department of cell biology*, http://kuhn.scripps.edu, 2012.

[17] Peter Kuhn and Paul Newton, *Bimonthly phone conversations and email correspondence*, personal communication, 2012.

[18] Debra Lacoste, *4. dendrograms (the "dendrogram" tab)*, http://publish.uwo.ca/~cantus/RSeries/R-Dendrogram.html, 2009.

[19] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce, *Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories*, Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (Washington, DC, USA), CVPR '06, IEEE Computer Society, 2006, pp. 2169–2178.

[20] David G. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision 60(2), 91110 (2004).

[21] Kanti Mardia, *Multivariate analysis*, Academic Press (1979).

[22] Marija Norusis, *Ibm spss statistcs guides: Cluster analysis*, http//www.norusis.com/pdf/SPC_v13.pdf, 2005-2012.

[23] E. Nowak, F. Jurie, and B. Triggs, *Sampling strategies for bag-of-features image classification*, Computer Vision–ECCV 2006 (2006), 490–503.

[24] S. O'Hara and B. Draper, *Introduction to the bag of features paradigm for image classification and retrieval*, arXiv preprint arXiv:1101.3354 (2011).

[25] Petra Perner, Horst Perner, and Bernd Müller, *Mining knowledge for hep-2 cell image classification*, Artificial intelligence in medicine **26** (2002), no. 1, 161–173.

[26] Mihalcea R. and Corley C., *Corpus-based and knowledge-based measures of text semantic similarity*, AAAI 06, p. 775780 (2006).

[27] A. Rakhlin and A. Caponnetto, *Stability of k-means clustering*, Advances in Neural Information Processing Systems **19** (2007), 1121.

[28] Conrad Sanderson and Brian C. Lovell, *Multi-region probabilistic histograms for robust and scalable identity inference*, Proceedings of the Third International Conference on Advances in Biometrics (Berlin, Heidelberg), ICB '09, Springer-Verlag, 2009, pp. 199–208.

[29] Yutaka Sasaki, *The truth of the f-measure*, Teaching, Tutorial materials, Version: 26th October (2007).

[30] Prajol Shrestha, *Corpus-based methods for short text similarity*, Rencontre des Étudiants Chercheurs en Informatique pour le Traitement automatique des Langues **2** (2011), no. 1.

[31] P. Soda and G. Iannello, *Aggregation of classifiers for staining pattern recognition in antinuclear autoantibodies analysis*, Information Technology in Biomedicine, IEEE Transactions on **13** (May), no. 3, 322–329.

[32] Sokal and Michener, *A statistical method for evaluating systematic relationships*, University of Kansas Science Bulletin 38: 1409 - 1438 (1958).

[33] Michael Steinbach, George Karypis, Vipin Kumar, et al., *A comparison of document clustering techniques*, KDD workshop on text mining, vol. 400, Boston, 2000, pp. 525–526.

[34] P. Strandmark, J. Ulen, and F. Kahl, *Hep-2 staining pattern classification*, Pattern Recognition (ICPR), 2012 21st International Conference on, Nov., pp. 33–36.

[35] Venkat Venkatasubramanian, *A review of process fault detection and diagnosis part i: Quantitative model-based methods*, Computers and Chemincal Engineering **27** (2003), 293–311.

[36] ———, *A review of process fault detection and diagnosis part iii: Process history based methods*, Computers and Chemincal Engineering **27** (2003), 327–346.

[37] Arnold Wiliem, Yongkang Wong, Conrad Sanderson, Peter Hobson, Shaokang Chen, and Brian C. Lovell, *Classification of human epithelial type 2 cell indirect immunofluoresence images via codebook based descriptors*, Applications of Computer Vision (WACV), 2013 IEEE Workshop on, Jan., pp. 95–102.

[38] Min Yu, Shannon Stott, Mehmet Toner, Shyamala Maheswaran, and Daniel A Haber, *Circulating tumor cells: approaches to isolation and characterization*, The Journal of cell biology **192** (2011), no. 3, 373–382.

A Appendix

A.1 Run1 Create Dictionary pictures only script

```
%% Tegan Emerson and David Hopkins
%main inputs:
% R,G,B slides

%key outputs:
% Panels with boundaries

%index:
%step 0: define parameters, preallocate variables
%step 1: read in a image, use edge detector to get cells, print image



%% Step 0: Define Parameters and Image information
clear, clc, close all
Dictionary_set_size = 35;  %(max 801) This is the number of images in our Dictionary set
num_words_in_dict = 400;

new_image_starting_idx = zeros(Dictionary_set_size,1);
%% step 1: read in a image, use edge detector to get cells, compute feature descriptor
 for each pixel in each ce

count = 1;
cdicount = 1;

count = 41545;
cdicount = 1;

for j = 1226
% for j = [ 69 201 226 273 277 301 355 515 533 543 556 584 629 640 727 740 750 752 ...
848 917 932 939 949 977 988 1118 1121 1139 1140 1163 1188 1214 1217 1226 1232 2273]
```

```
    new_image_starting_idx(j) = count;
%     tic, close all

    Ired = (imread(['/data3/emerson/newdata/slideid_007072/Tile00'...
    ,num2str(j+2304),'.tif']));
    Igreen =(imread(['/data3/emerson/newdata/slideid_007072/Tile00'...
    ,num2str(j+4608),'.tif']));
    Iblue = (imread(['/data3/emerson/newdata/slideid_007072/Tile00'...
    ,num2str(j),'.tif']));

    IComposite(:,:,1)=2*Ired;
    IComposite(:,:,2)=Igreen;
    IComposite(:,:,3)=Iblue;

    BW = im2bw(IComposite, .12); % graythresh(IComposite)); %OTSU (1979)
    B = bwboundaries(BW,4,'noholes'); %labels: L.  %B:px1 cell array

    fig1 = figure(j);
    imshow(IComposite,'Border','tight'); set(fig1,'Position',[1 1 800 800]);

    for i = 1:length(B)
        boundary = B{i};

        minx = max(6,min(boundary(:,2)));
        maxx = min(max(boundary(:,2)),1356);
        miny = max(6,min(boundary(:,1)));
        maxy = min(max(boundary(:,1)),998); %maxx or maxy should not be on the...
         boundary because these pixels were removed, then we also keep in mind...
          we are sending a 4 pixel boundary...
         around each cell to the descriptor function

        %filter out noise (stuff less than n pixels in any direction)
        if maxx-minx < 2 || maxy - miny < 2
            continue;
```

```
        elseif maxx-minx >= 1360 || maxy-miny >= 1002 %too large
            continue;
        end

        figure(j), hold on, plot(boundary(:,2), boundary(:,1), 'c', 'LineWidth', 2)
        text( mean([minx maxx]), mean([miny maxy]),...
        strtrim(cellstr(num2str((count)'))),'FontSize',8,'color','y');

        count = count+1 %update to next cell
    end

%     saveas(fig1,['Composite Image ',num2str(j)], 'png')
%     pause
%     comp_times(j) = toc;
end

% total_time = cumsum(comp_times);
% total_time = total_time(end);
% disp(['Step 1 finished in : ', num2str(total_time/60), ' minutes'])
```

A.2 func image gradient MAG and TH script

```
function [mag,th] = func_image_gradient_MAG_and_TH(X)

    dx = [1, -1]; %???use a sobel gradient patch
     so as not to shift image by half a pixel
    dy = [1; -1];
    Ix = conv2(single(X),dx,'valid'); %compute image gradient in x direction
    Iy = conv2(single(X),dy,'valid');
    %valid region is:
    Ix = Ix(1:size(Iy,1),1:size(Ix,2));
    Iy = Iy(1:size(Iy,1),1:size(Ix,2));
```

```
    mag = sqrt(Ix.*Ix + Iy.*Iy); %magnitude of gradients
    th = atan2(Iy,Ix);% + pi;
```

A.3 func feature descriptor oriented by pixel ii

```
function out = func_feature_descriptor_oriented_by_pixel_ii(TH,MAG)

[mint,nint] = size(TH);
Cell_descriptors = cell(mint-8,nint-8);

for i=5:mint-4 % only zip through the interior cells
    for j=5:nint-4


        %we'll only consider the 9x9 grid around a particular pixel.
        %everytime you see the entry (5,5), that's the center pixel, the
        %one for which we're constructing a feature descriptor.
        THij  = TH(i-4:i+4,j-4:j+4);
        MAGij = MAG(i-4:i+4,j-4:j+4);

        th0 = THij(5,5);



        %find all thetas that fall in regions 1 - 4
        Region_1 = THij>= mod(th0-.25*pi,2*pi) & THij<mod(th0+.25*pi,2*pi);
        Region_2 = THij>= mod(th0+.25*pi,2*pi) & THij<mod(th0+.75*pi,2*pi);
        Region_3 = THij>= mod(th0+.75*pi,2*pi) & THij<mod(th0+1.25*pi,2*pi);
        Region_4 = THij>= mod(th0+1.25*pi,2*pi) & THij<mod(th0+1.75*pi,2*pi);



        clear MAGx4
        MAGx4 = zeros(4,9,9);
        MAGx4(1,:,:) = Region_1.*MAGij;
```

```
    MAGx4(2,:,:) = Region_2.*MAGij;
    MAGx4(3,:,:) = Region_3.*MAGij;
    MAGx4(4,:,:) = Region_4.*MAGij;

    %build cell array of magnitudes based on cardinal directions for each pixel
    Cij = num2cell(MAGx4,1);
    Cij =  reshape(Cij,9,9);


    %build 36X1 vector for the center pixel based on its 3x3 grid
    mm = 2; nn = 2; %add up vectors from topleft 3x3 grid


C3vec(:,1) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1))...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 2;  nn = 5;%add up vectors from topcenter 3x3 grid

C3vec(:,2) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
        +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 2;  nn = 8;
C3vec(:,3) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 5;  nn = 2;
C3vec(:,4) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
```

```
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 5;  nn = 5;    C3vec(:,5) =...
   cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
   cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 5;  nn = 8;
C3vec(:,6) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 8;  nn = 2;
C3vec(:,7) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 8;  nn = 5;
  C3vec(:,8) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
  cell2mat(Cij(mm-1,nn+1))...
      +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
      +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));

  mm = 8;  nn = 8;
C3vec(:,9) = cell2mat(Cij(mm-1,nn-1))+cell2mat(Cij(mm-1,nn))+...
cell2mat(Cij(mm-1,nn+1))...
     +cell2mat(Cij(mm  ,nn-1))+cell2mat(Cij(mm  ,nn))+cell2mat(Cij(mm  ,nn+1)) ...
     +cell2mat(Cij(mm+1,nn-1))+cell2mat(Cij(mm+1,nn))+cell2mat(Cij(mm+1,nn+1));
   Cell_descriptors(i-4,j-4) = {C3vec(:)'};
```

```
    end
end



out = Cell_descriptors;
```

A.4 Run1 Create Dictionary script

```
%% Tegan Emerson and David Hopkins
%main inputs:
% R,G,B slides

%key outputs:
% C400 - Dictionary with 400 words

% Cell_descriptor_vec - unabridged dictionary
% Cell_images
% image with boundaries and cell numbering



%index:
%step 0: define parameters, preallocate variables
%step 1: read in a image, use edge detector to get cells, compute feature
%descriptor for each pixel in each cell
%step 2: K-means to get dictionary C400



%% Step 0: Define Parameters and Image information
clear, clc, close all
Dictionary_set_size = 36;  %(max 801) This is the number of images in our Dictionary set
num_words_in_dict = 400;

est_number_of_cells = Dictionary_set_size * 2000; %over estimation of number of cells
```

```
% preallocate variables
Cell_descriptor_vec = zeros(400*est_number_of_cells,108); %over estimation
%
% Cell_descriptor_vec = spalloc(400*est_number_of_cells,108,Dictionary_set_size*10^7);
new_image_starting_idx = zeros(Dictionary_set_size,1);

comp_times = zeros(Dictionary_set_size,1);

%% step 1: read in a image, use edge detector to get cells, compute feature ...
 descriptor for each pixel in each ce

count = 1;
cdicount = 1;

for j = [69 201 226 273 277 301 355 515 533 543 556 584 629 640 727 740 750 752 848...
 917 932 939 949 977 988 1118 1121 1139 1140 1163 1188 1214 1217 1226 1232 2273]
    new_image_starting_idx(j) = count;
    tic

    Ired = imread(['/data3/emerson/colorlayers/','redlayer (',num2str(j),').tif']);
    Igreen =imread(['/data3/emerson/colorlayers/','greenlayer (',num2str(j),').tif']);
    Iblue = imread(['/data3/emerson/colorlayers/','bluelayer (',num2str(j),').tif']);
    IComposite(:,:,1)=2*Ired;
    IComposite(:,:,2)=Igreen;
    IComposite(:,:,3)=Iblue;

    [mag_red, th_red]=func_image_gradient_MAG_and_TH(Ired);
    [mag_green, th_green]=func_image_gradient_MAG_and_TH(Igreen);
    [mag_blue, th_blue]=func_image_gradient_MAG_and_TH(Iblue);

    BW = im2bw(IComposite, .12); % graythresh(IComposite)); %OTSU (1979)
    B = bwboundaries(BW,4,'noholes'); %labels: L.  %B:px1 cell array

    for i = 1:length(B)
```

```
boundary = B{i};

minx = max(6,min(boundary(:,2)));
maxx = min(max(boundary(:,2)),1356);
miny = max(6,min(boundary(:,1)));
maxy = min(max(boundary(:,1)),998); %maxx or maxy should not be on the...
 boundary because these pixels were removed, then we also keep in mind...
 we are sending a 4 pixel boundary around each cell to the descriptor function

% filter out noise (stuff less than n pixels in any direction)
if maxx-minx < 2 || maxy - miny < 2
    continue;
elseif maxx-minx >= 1360 || maxy-miny >= 1002 %too large
    continue;
end

%crop number corresponds to color, and here the crop numbers
%correspond to 1=red 2=green 3=blue
 clear cdivec_red cdivec_green cdivec_blue cdivec

Cdi_red = func_feature_descriptor_oriented_by_pixel_ii(th_red(miny-4:maxy+4,...
    minx-4:maxx+4),mag_red(miny-4:maxy+4,minx-4:maxx+4));
    cdivec_red = cell2mat(Cdi_red(:));

Cdi_green = func_feature_descriptor_oriented_by_pixel_ii(th_green(...
miny-4:maxy+4, minx-4:maxx+4),mag_green(miny-4:maxy+4,minx-4:maxx+4));
    cdivec_green = cell2mat(Cdi_green(:));

Cdi_blue = func_feature_descriptor_oriented_by_pixel_ii(th_blue(...
miny-4:maxy+4, minx-4:maxx+4),mag_blue(miny-4:maxy+4,minx-4:maxx+4));
    cdivec_blue = cell2mat(Cdi_blue(:));

    cdivec= [cdivec_red cdivec_green cdivec_blue];
```

```
    Cell_descriptor_vec(cdicount:cdicount+size(cdivec,1)-1,:) = cdivec; %concatenate...
        feature descriptors from current cell.

        cdicount = cdicount+size(cdivec,1);
         %update by number of feature descriptors concatenated
        count = count+1 %update to next cell
    end

    comp_times(j) = toc;
end

total_time = cumsum(comp_times);
total_time = total_time(end);
disp(['Step 1 finished in : ', num2str(total_time/60), ' minutes'])


%% Step 2: K-means feature descriptor vector and distortion error plots
%num_words_in_dict = 400;
tic
 Cell_descriptor_vec(all(Cell_descriptor_vec==0,2),:)=[]; %removes the all-zero...
  feature descriptor rows number_feature_descriptors=size(Cell_descriptor_vec,1);
 distortion_error_information=zeros(1500,2);
 for i=10 %[10 50 200 400 600 800 1000 1200 1500]
    distance_between_features_and_nearest_center=zeros(number_feature_descriptors,1);
 [IDX,Dictionary] = K-means(Cell_descriptor_vec,i,'distance','cosine',...
 'start','cluster');
    save(['Dictionary ','num2str(i)'], Dictionary)
 for j=1:i
     feature_descriptors_with_index_j=find(IDX==j);
     for k=feature_descriptors_with_index_j
         distance_between_features_and_nearest_center(k)=
         sum((Cell_descriptor_vec(k)'-Dictionary(j)').^2);
     end
 end
```

```
 distortion_error_information(i,1)=i;
 distortion_error_information(i,2)=sum(distance_between_features_and_nearest_center);

 end



 toc
 disp(['Step 2 finished in : ', num2str(toc/60), ' minutes'])
```

A.5 Run2 Build Term Vectors script

```
%% input: C400 - feature descriptors Kmean centers

%% preallocate variables
est_num_cells = 4378;  % estimated number of cells
num_prebuckets = 30;  %find on distortion graph
Training_Term_Vec_Set_size = 100; %66;
rotation_angle = 0; % x90 degrees



num_words_in_dict = size(C400,1);

terms = zeros(est_num_cells,num_words_in_dict);
 %guessing we won't see more than 3000 cells for our ...
current database.  this can change of course.
Cell_image_boundaries = cell(est_num_cells,1);

% new_image_starting_idx = zeros(Training_Term_Vec_Set_size,1);
Icrop = zeros(250,250,4);
mag = zeros(249,249,4);
th= zeros(249,249,4);
comp_times = zeros(Training_Term_Vec_Set_size,1);
```

```
Image_croppings = [1:250; 251:500; 501:750; 751:1000];
count = 1;
%%
for j = 1:Training_Term_Vec_Set_size
%     new_image_starting_idx(j) = count;
    tic
%     close all
  I = imread(['/home/katrina/b/hopkins/Desktop/Scripps/data/Dictionary_Set/','panel
   (',num2str(j),').png']);

   imread(['/home/katrina/b/hopkins/Desktop/Scripps/data/Training_Term_Vector_Set/'...
   ,'panel (',num2str(j),').png']);

    for layer_num = 1:3
    IComposite(:,:,layer_num) = rot90( I(1:250,1001:1250,layer_num) , rotation_angle );
    end

    for crop_num = 1:4
   Icrop(:,:,crop_num)=rot90( I(1:250,Image_croppings(crop_num,:),1), rotation_angle );
            % compute the gradients, magnitudes, and thetas, for each frequency
  [mag(:,:,crop_num), th(:,:,crop_num)]=...
  func_image_gradient_MAG_and_TH(Icrop(:,:,crop_num));
    end


    BW = im2bw(IComposite, .15); % graythresh(IComposite)); %OTSU (1979)
    B = bwboundaries(BW,4,'noholes'); %labels: L.  %B:px1 cell array


    for i = 1:length(B)
        boundary = B{i};

        minx = max(6,min(boundary(:,2))); maxx = min(max(boundary(:,2)),244); miny = ...
```

```
        max(6,min(boundary(:,1))); maxy = min(max(boundary(:,1)),244);
        %maxx or maxy should not be on the ...
        boundary because these pixels were removed,...
         then we also keep in mind we are sending a 4 pixel ...
        boundary around each cell to the descriptor function

        % filter out noise (stuff less than n pixels in any direction)
        if maxx-minx < 5 || maxy - miny < 5
            continue;
        elseif maxx-minx >= 248 || maxy-miny >= 248 %too large
            continue;
        end

        Cell_image_boundaries(count,1) = {boundary};
        Cell_image_boundaries(count,2) = {j};

        cdivec = zeros((maxx-minx+1)*(maxy-miny+1),144);
        for crop_num = 1:4
%  Cell_images_frequencies(count,crop_num) = {Icrop(miny:maxy,minx:maxx,crop_num)};
      Cdi = func_feature_descriptor_oriented_by_pixel_ii(...
        th(miny-4:maxy+4,minx-4:maxx+4,crop_num),...
            mag(miny-4:maxy+4,minx-4:maxx+4,crop_num));
            cdivec(:,(crop_num-1)*36 + (1:36)) = cell2mat(Cdi(:));
        end

%     figure(j), hold on, plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2)
%     text( mean([minx maxx]), mean([miny maxy]),...
strtrim(cellstr(num2str((count)'))), 'color','y',...
'Fontsize',12,'FontWeight','bold');


        %build term vector - slick coding
        [minval,minidx] = min(pdist2(cdivec,C400,'cosine'),[],2);
        %minidx tells us the number of times a certain word appears!
```

```
        clear cdivec Cdi

        for iterm = 1:num_words_in_dict
            terms(count,iterm) = sum(sum(minidx==iterm));
        end
        terms(count,:) = terms(count,:) / ((maxx - minx) * (maxy - miny));
        % scale down term vector...
         by the size of the cell!

        count = count+1;
    end
%     saveas(fig1,['Composite Image ',num2str(j),' Rotated ',...
num2str(rotation_angle),'x90 degrees'], 'png')
    comp_times(j) = toc
end

terms(all(terms==0,2),:)=[]; %removes the all-zero rows
```

A.6 Run3 Analyze Term Vector Clusters script

```
%% Analyze Term Vectors
% inputs:
%     load - RUN2_results


%parameters to optimize:
%     num_buckets: line 50


%items to calculate by hand:
%     idx_buckets
```

```
% Code Outline
% 1.  K-means of Term Vectors to get prebuckets
% 2.  Hierarchical Cluster the prebuckets, then manualy assign the...
 prebucket idx to each bucket
% 3.  plot panels with colored boundaries
% 4.  Bar Graphs with CTC bucket assignments overlayed

%% 1.  K-means of Term Vectors to get prebuckets
% rand('seed',11)
%
% [idx_term2prebucket_k,prebuckets] = K-means(terms,30,'distance',...
'cosine','start','cluster');
% % [idx_term2prebucket_k,prebuckets,sumd,D]
% zprebuckets = linkage(prebuckets,'average','cosine');


%% 1.5 for Rotated Terms
% [minval,idx_3x90degrees] = min(pdist2(terms,prebuckets,'cosine'),[],2);
 % minidx tells us the number...
 of times a certain word appears!


%% 2.
% figure, stem(  sum(terms,1) / size(terms,1)  ,'b')
% hold on
% stem(  sum(terms(CTC_cell_index,:),1) / length(CTC_cell_index)  ,'r')
% legend('all terms','CTCs','Location','NorthWest')
% xlabel('Word no. in Term vector')
% ylabel('Frequency of word occurence')
% title('Term Vector Analysis: CTCs vs All Cells')

% figure, stem(  sum(prebuckets,1) / size(prebuckets,1)  ,'b')
% hold on
```

```
% stem(  sum(prebuckets(30,:),1) / size(prebuckets(30,:),1)  ,'r')
% legend('others','outlier','Location','NorthWest')
% xlabel('Word no. in Term vector')
% ylabel('Frequency of word occurence')
% title('Prebucket Term Vector Analysis: outlier bucket vs others')

%% 2.  Hierarchical Cluster the prebuckets,...
 then manualy assign the prebucket idx to each bucket

num_buckets = 6; %3 or 6;  desired number of buckets

if num_buckets == 3;
    colorthresh = .5;
elseif num_buckets == 6;
    colorthresh = .35;
end



figure, [h, TT, perm] = dendrogram(zprebuckets,'colorthreshold',colorthresh);
set(h,'LineWidth',2);
handles_dendro_new = findobj(h,'flat','color','r');
set(handles_dendro_new,'color',[1 0 1])
grid on
xlabel('Prebucket number')

% indexes the prebuckets to their bucket
if num_buckets == 3
        idx_buckets{1} = perm(1:23);
        idx_buckets{2} = perm(24:29);
        idx_buckets{3} = perm(30);
        title('3 bucket hierarchical clustering')
end
if num_buckets == 6
        idx_buckets{1} = perm(1:11);
```

```
        idx_buckets{2} = perm(12:16);
        idx_buckets{3} = perm(17:23);
        idx_buckets{4} = perm(24:27);
        idx_buckets{5} = perm(28:29);
        idx_buckets{6} = perm(30);
        title('6 bucket hierarchical clustering')
end


    %% 3.  plot panels with colored boundaries

    %
    %
    % for jpanel = 1:4;  %1:10;  %1:10 - shows the panels 1 to 10.
    % I = imread(['/home/katrina/b/hopkins/Desktop/Scripps/data/Dictionary_Set/',...
    'panel (', num2str(jpanel),').png']);
    %     IComposite = I(1:250,1001:1250,:);
    % fig1 = figure(jpanel); imshow(IComposite,'Border','tight');
     set(fig1,'Position',[200 800 800 800]);
    %     cells_in_jth_panel = cell2mat(Cell_image_boundaries(:,2)) == jpanel;
    %
    %     for ibucket = 1:num_buckets
    %         cells_in_ith_bucket = ismember(idx_term2prebucket_k,idx_buckets{ibucket});
    %
    %         clear boundaries
    %     boundaries = Cell_image_boundaries(cells_in_jth_panel &...
     cells_in_ith_bucket,:);
    %         for k = 1:size(boundaries,1)
    %             boundary = boundaries{k,1};
    %         figure(jpanel), hold on, plot(boundary(:,2), boundary(:,1), 'Color',  ...
    dendro_colors{ibucket}, 'LineWidth', 3)
    %         end
    %     end
    %     saveas(fig1,['Composite Image ',num2str(jpanel),' with Borders'], 'png')
```

```
    %
    % end


    %% 4.  Bar Graphs with CTC bucket assignments overlayed



    clear Nb Nctc Nctcwbc Nrn Nrnwbc
    for ibucket = 1:num_buckets;
        cells_in_ith_bucket = ismember(idx_term2prebucket_k,idx_buckets{ibucket});
        ctcs_in_ith_bucket = cells_in_ith_bucket(CTC_cell_index);
        ctcs_with_wbc_in_ith_bucket = ...
         cells_in_ith_bucket(CTC_with_WBC_cell_index);
        rednoise_in_ith_bucket = cells_in_ith_bucket(Red_noise_cell_index);
     rednoise_with_wbc_in_ith_bucket = cells_in_ith_bucket(Red_noise_with_WBC_cell_index);



        Nb(ibucket) = sum(cells_in_ith_bucket); %number of cells in ith bucket
        Nctc(ibucket) = sum(ctcs_in_ith_bucket); %number of ctc's in ith bucket
        Nctcwbc(ibucket) = sum(ctcs_with_wbc_in_ith_bucket);
        Nrn(ibucket) = sum(rednoise_in_ith_bucket);
        Nrnwbc(ibucket) = sum(rednoise_with_wbc_in_ith_bucket);


    end


figure, hold on
mydata = [Nctc' Nctcwbc' Nrn' Nrnwbc'];
bar_h = bar(mydata);
bar_children=get(bar_h,'Children');

%change color
set(bar_children{1},'FaceColor','r');
set(bar_children{2},'FaceColor','r','EdgeColor','g','LineWidth',2);
```

```
set(bar_children{3},'FaceColor','b');
set(bar_children{4},'FaceColor','b','EdgeColor','g','LineWidth',2);

%change legend to match color
for i = 1:4
  set(get(get(bar_h(i),'Annotation'),...
  'LegendInformation'),'IconDisplayStyle','Children');
end

legend('CTC','CTC w/ wbc', 'red noise','red noise w/ wbc','Location','NorthWest')
grid on
ylabel('no. of occurences'), xlabel('Bucket no.')

%add text
x_loc = get(bar_h,'XData');
x_loc = x_loc{1};
x_loc_shift = [-.28 -.07 .07 .274]; %values to shift x cordinates of text labels

for k=1:4
text(x_loc+x_loc_shift(k),mydata(:,k)',num2str(mydata(:,k)),...
    'HorizontalAlignment','center',...
    'VerticalAlignment','bottom');
end

for kk = 1:6
text(x_loc(kk), max(mydata(kk,:))+3,['/',num2str(Nb(kk))],...
    'HorizontalAlignment','center',...
    'VerticalAlignment','bottom');
end

axis([x_loc(1)-.9 x_loc(end)+.9 0 max(max(mydata))+6])

title('Occurence counts of various cell types per bucket')
```

A.7 secondpass script

```
PD =  pdist2(terms,prebuckets,'cosine');
[minval,minidx] = min(PD,[],2);

 for i = 1:length(terms)
     PD(i,minidx(i)) = 1;
 end

[minval,minidx2] = min(PD,[],2);
second_prebucket_class = minidx2;
```

B Glossary

B.1 Mathematics: in order of ideas

- **event**: region of connected pixels determined by edge detection and distinct from the base background value.
- **feature descriptor**: Vector of local curvature information associated with each pixel
  length: 36
- **cluster**: a grouping of data points with inherent similarities
- **cluster center**: the single mean point amongst data points within one cluster.
- **K-means**: a iterative, centroid-based clustering technique which minimizes the distance between datapoints and their assigned cluster center. K centroids exist a priori.
- **dictionary word**: the K-means centroid of a cluster of feature descriptors
  length: 36
- **dictionary**: the set of the n "most important" words the centroids that reduce the sum of squared errors with amongst feature descriptors and their closest centroids to the elbow point.
- **term vector**: a vector of word frequencies associated with every event
  length: n. if n is large, then the term vector is sparse
- **prebucket**: Center of a Term Vector K-means cluster Represented by the cell image closest to cluster center
- **hierarchical clustering**: a connectivity based clustering technique that groups cells by relating their distance amongst data points. Two data points are considered similar if their pairwise distance is smaller.
- **dendrogram**: a figure designed to display the results of hierarchical clustering on a set of data points.
- **bucket**: Cluster of prebuckets as determined by a hierarchical clustering. Represents a classification type of cells
- **Distortion Error**: Sum of residual distances between every data point and its nearest cluster center. Used to determine appropriate size of dictionary and number of Pre-Buckets
- **elbow plot**: see Distortion Error

B.2 Mathematics: in alphabetical order

- **bucket**: Cluster of prebuckets as determined by a hierarchical clustering. Represents a cell-subtype
- **cluster**: a grouping of data points with inherent similarities
- **cluster center**: the single mean point amongst data points within one cluster.
- **dendrogram**: a figure designed to display the results of hierarchical clustering on a set of data points.
- **dictionary**: the set of the n "most important" words the centroids that reduce the sum of squared errors with amongst feature descriptors and their closest centroids to the elbow point.
- **dictionary word**: the K-means centroid of a cluster of feature descriptors
  length: 36
- **distortion error**: Sum of residual distances between every data point and its nearest cluster center. Used to determine appropriate size of dictionary and number of Pre-Buckets
- **elbow plot**: see distortion error
- **event**: region of connected pixels determined by edge detection and distinct from the base background value.
- **feature descriptor**: Vector of local curvature information associated with each pixel
  length: 36
- **hierarchical clustering**: a connectivity based clustering technique that groups cells by relating their distance amongst datapoints. Two data points are considered similar if their pairwise distance is smaller.
- **K-means algorithm**: a iterative, centroid-based clustering technique which minimizes the distance between data points and their assigned cluster center. K centroids exist a priori.
- **prebucket**: Center of a Term Vector K-means cluster. Represented by the cell image closest to cluster center
- **term vector**: a vector of word frequencies associated with every event.
  length: n. if n is large, then the term vector is sparse