Unsupervised brightfield image segmentation with RPCA and spectral clustering

by

Yuehuan Chen

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Justin W.L.Wan

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this report. This is a true copy of the report, including any required final revisions, as accepted by my examiners.

I understand that my report may be made electronically available to the public.

Abstract

Microscopic image analysis is an important step in biological study. Biologists study the movements of cells under certain drug treatments through a sequence of time-lapse microscopic images to determine the effect of the treatments. The development of modern bright-field microscopes allows more detailed investigations of the cell activities. However, it also brings challenges for automatic cell image analysis because of the low-contrast nature of bright-field microscopic images.

This paper presents contributions to automatic bright-field cell image segmentation. We propose ten methods for bright-field cell image segmentation. The ten methods are based on two well-known methods in computer vision, namely spectral clustering and robust principal component analysis (RPCA). The first three methods are based on spectral clustering. They determine the segmentations by classifying the k segments from spectral clustering into cell segments and background segments. The other three methods are RPCA-based methods. The cell image segmentation problem is treated as a background subtraction problem, where the cells are the sparse moving objects to be identified. Several modifications have been made to RPCA to improve the segmentations. In the last four methods, we combine spectral clustering and RPCA to solve the segmentation problem. The first two methods use the results from RPCA to help the segmentation based on spectral clustering. In the last two methods, we formulate the problem as a principal component pursuit with graph cut penalization, and obtained the segmentation results similar to the three RPCA-based methods presented previously. The last method outperforms all previous methods in terms of segmentation quality.

We have applied these methods on a set of C2C12 cells in bright-field microscopy. Experimental results confirm that the proposed methods give accurate segmentation of cells in bright-field microscopy, which conventional image segmentation methods cannot attain.

Acknowledgements

I would like to thank everyone who has helped me during my graduate study at the University of Waterloo. Particular thanks go to my supervisor Prof. Justin W.L. Wan. Without his valuable advice and firm support, this work would not have been possible. I also want to thank Prof. Yuying Li for her help in reading this paper, and her advice on making this paper better. Last but not least, I would like to thank all of my colleagues in the Computational Mathematics Master Program. Without their help and encouragements, my master study would not have been well-spent.

Dedication

This is dedicated to my parents and my brother.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

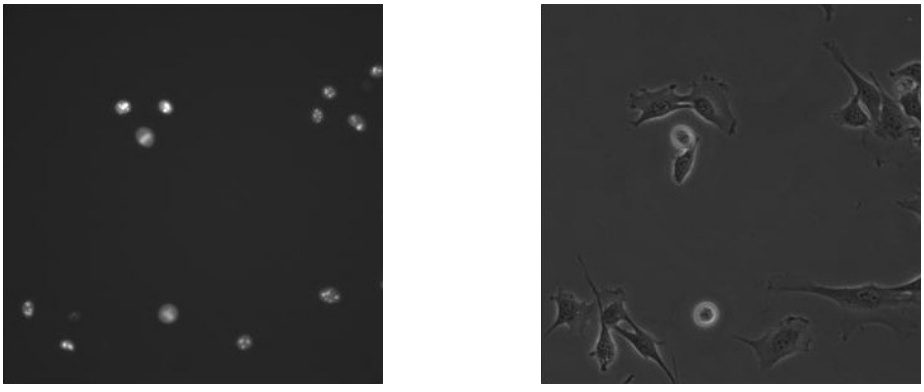
Image segmentation, a well-developed field in computer vision, describes the process of partitioning an image into more semantic segments. It is the first fundamental step in many applications, for example, tracking a moving object in a video surveillance and recognizing different types of objects in a vision guided assembly system. Its applications range from medical imaging to traffic control systems. The result of image segmentation is a simplification of the original image from tremendous number of pixels to just a few segments, and thus provides additional information to aid further analysis and higher level processing.

In many cases, segmentation reduces the problem of identifying the boundaries of the foreground region. More specifically, it becomes a task of classifying pixels into foreground pixels and background pixels. This task can be performed easily in human vision system. However, the massive amount of images from laboratories necessitates the use of computer to facilitate the segmentation. Various methods have been introduced in literature on image segmentation. However, there is not any general method gives good results in all applications. Moreover, some method is more desirable than the others in particular applications, because we are looking for different forms of segmentation in different applications. For example, in object recognition, the objective is to capture the basic shape of the objects. However, in the detection of brain lesions, a more precise segmentation of the white matter of the brain is required. In addition, pixel intensities and textures vary slightly in the images, which complicate the segmentation process. One particular challenge in this area is the automatic segmentation of eukaryotic cells in bright-field microcopies.

1.1 Bright-field Cell Images

The motivation for this project comes from its applications in medical image segmentations for eukaryotic cells. Eukaryotic cells are complex cells that are part of plants and animals. These cells are transparent and contain different organelles in their cytoplasm. The most fundamental characteristic of Eukaryotic cells, which makes them different from prokaryotic cells, is that they have nuclei, within which the genetic material like deoxyribonucleic acid (DNA) is contained. The presence of a nucleus differentiates the cell division process of eukaryotic cells and prokaryotic cells. During its reproduction cycle, a eukaryotic cell goes through five stages of cell division, replicating its DNA and all of its internal organelles in this process.

In an attempt to capture the many phases of cell division for eukaryotic cells, biologists have adopted a few different methods of microscopy. Two of the most popular kinds are fluorescent microscopy and bright-field microscopy. Fluorescent microscopy is a more complicated method done by introducing a gene to the cell so that it emits a certain light around its nucleus. This light makes the segmentation process very easy, but the images often only capture the nucleus of the cell where the DNA is contained. It is also possible that the gene introduced in fluorescent microscopy affects the behaviour of the cell during cell division. In this case, biologists turn to bright-field microscopy, where the images provide information on the cytoplasm as well as the nucleus. Figure 1.1 compares the fluorescent microscopy and the bright-field microscopy. The fluorescent images only show the nuclei of the cells, but the bright-field images show the complete cells.



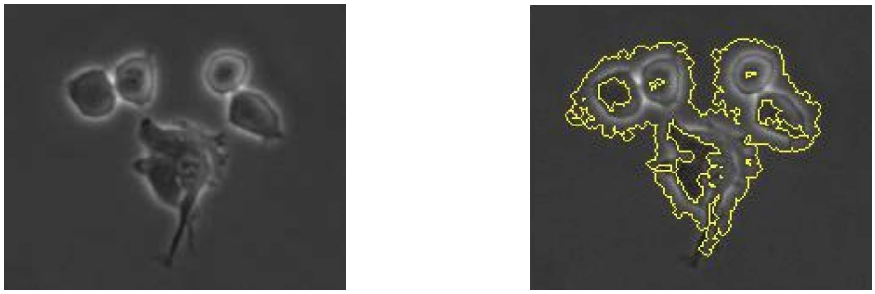
(a) (b)

Figure 1.1: Comparison of fluorescent microscopy and bridght-field microscopy

However, in bright-field microscopy, the overall intensity of the cell is very similar to that of the background, hence making it difficult to perform image segmentation. In these images, different regions of a cell absorb different degrees of light, which makes some regions brighter than the others. This makes the cell body not uniform in intensity level and many algorithms have difficulty identifying where the cell body is. Furthermore, the method of bright-field microscopy introduces a halo around the cell, but the halo may not be consistent. This halo causes difficulty in edge detection algorithms, as the halo is not consistent throughout different stages of the cell division. Image segmentation applied to bright-field microscopy is therefore a much more difficult problem compared to its fluorescent counterpart.

1.2 Bright-field Image Segmentation

Various image segmentation techniques have been tried to segment cell images. The most classical method is the edge evolution method. It begins with an initial curve within the image boundary, and moves the curve according to an equation defined by an energy functional, which is related to the pixel intensity inside the cells and outside the cells. It is defined in a way such that the evolving curve is stopped at the cell boundaries. However, this type of method is not successful in segmenting bright-field cell images. In particular, only the halo has been identified as foreground in most cases. Figure 1.2 shows the segmentation result of applying a curve evolution method[1] on the bright-field cell images. The evolving curve can only capture the halo, because the pixel intensities inside the cell bodies are indistinguishable from the background.



(a) (b)

Figure 1.2: Segmentation of bright-field image using Chan-Vese

[1]We have applied the Chan-Vese active contour without edge algorithm on the cell image.

Spectral clustering has gained in popularity recently on image segmentation, and it has been applied on segmenting bright-field cell images [2]. However, the method can only produce a decent segmentation for images with only a couple of cells. Therefore, a global-local method was proposed in [2]. The method first segments an individual cell cluster in a confine area, and then combines every piece of segmented cell cluster to be the final segmentation result. Figure 1.3 shows the segmented cells using the spectral clustering method in [2]. This method gives a good segmentation result, but the method is hard to implement. Each individual cell cluster is extracted by identifying its location manually. The goal of this project is to develop some robust mathematical models to segment a bright-field cell image without human intervention.

Figure 1.3: Spectral clustering on an image with one cell cluster

The remainder of this research paper is organized as follows. Chapter 2 will introduce some background knowledge of spectral clustering, robust principal component analysis (RPCA), and Chan-Vese active contour without edge model. They are the main methods we will use in the image segmentation algorithms. In Chapter 3, we will introduce ten methods. The first 3 methods are based on spectral clustering. The others are based on RPCA, but spectral clustering are also involved in the last 4 methods. In Chapter 4, we apply the methods on a set of bright-field cell images, and display the segmentation results. Finally, we will end our discussion by some concluding remarks.

Chapter 2

Background

In this Chapter, we give an overview on some existing methods in the field of computer vision. These methods are adopted in the methods proposed in Chapter 3. The cell segmentation methods proposed in Chapter 3.1, and Chapter 3.3 use spectral clustering, which we will discuss in Chapter 2.2. For a given sequence of microscopic images, we have developed several cell segmentation methods based on a data analysis model called robust principal component analysis (RPCA) in Chapter 3.2 and Chapter 3.3. The model of RPCA will be reviewed in Chapter 2.3. In some RPCA based methods proposed in Chapter 3, classical image segmentation techniques are used as the post-processing steps. We will briefly talk about the classical image segmentation techniques in Chapter 2.1.

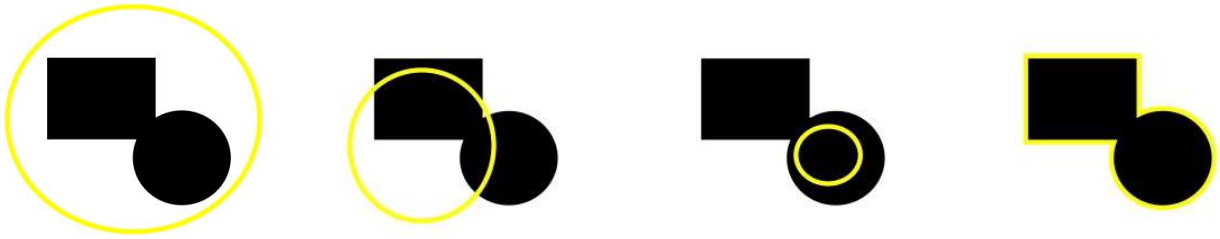
2.1 Classical Image Segmentation Methods

Most image segmentation algorithms assume that pixels belonged to the same object are connected by local grouping cues, for example pixel intensities, pixel location, and texture. In the simplest case, global attribute is sufficient in segmenting an image. Methods use solely global attribute are called non-contextual methods. One typical example of a non-contextual method is intensity-based thersholding. This simple method does not exploit enough information from the image to produce a good segmentation result. However, it is a useful tool for improving the segmentation quality if the thresholds can be obtained from other more sophisticated methods. Contextual methods, on the other hand, consider local attributes. They group pixels with similar attributes together to form the segmentation result. Region-growing methods, edge-based methods, watershed segmentation methods,

and active contour methods are all contextual methods. The Chan-Vese model is an example of region based active contour model. It begins with an initial contour,and evolves the contour according to an evolution equation in such a way that it stops on the object boundaries. More specifically, if we let u be the given image, and C be the evolving curve. The Chan-Vese model achieves segmentation according to the following energy functional:

$$F(c_1, c_2, C) = \underbrace{\mu Length(C) + \upsilon Area(inside(C))}_{regularizing\,terms} + \underbrace{\lambda_1 \int_{inside(C)} \mid u(x,y) - c_1 \mid dxdy + \lambda_2 \int_{outside(C)} \mid u(x,y) - c_2 \mid dxdy,}_{fitting\,terms} \tag{2.1}$$

where c_1 is the average intensity value inside the contour C, c_2 is the average intensity outside the contour C, $inside(C)$ is the region inside the curve C, and $outside(C)$ is the region outside the curve C. The regularity part is to control the smoothness of the contour C, while the fitting part is to move the contour to object boundaries. Figure 2.1 illustrate how this model works.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

Figure 2.1: All possible cases in the position of the curve: the fitting term is minimized in the case when the curve and the object boundary coincide. Case 1: $F_1(C) > 0, F_2(C) \approx 0 \Rightarrow Fitting > 0$. Case 2: $F_1(C) \approx 0, F_2(C) > 0 \Rightarrow Fitting > 0$. Case 3: $F_1(C) > 0, F_2(C) \approx 0 \Rightarrow Fitting > 0$. Case 4: $F_1(C) \approx 0, F_2(C) \approx 0 \Rightarrow Fitting = 0$.

By minimizing the fitting terms, we can minimize the difference between the intensities in $inside(C)$ and the foreground, and the difference between the intensities in $outside(C)$ and the background, and thus move the contour C to the actual object boundaries C_0.

2.2 Spectral Clustering and Image Segmentation

Clustering is a procedure of collecting a set of data points together so that the data points in the same group are similar and data points in different groups are dissimilar. It is a crucial step in data analysis, and a well-explored area in data mining. The most popular existing algorithms include k-means clustering, the Gaussian mixture model, and spectral clustering. In the problem of image segmentation, spectral clustering is usually the most effective algorithm among the others, because spectral clustering uses a user-defined similarity measurement. Therefore, the most interesting features of pixels such as pixel intensities, RGB values, and pixel locations can be focused more on when performing the clustering.

To apply spectral clustering in image segmentation, the given image is transformed into a weighted undirected graph $G = (V, E)$, where each node corresponds to a pixel and each edge is formed between each pair of pixels. Weights are assigned to all edges to reflect the image structure. If two pixels belong to a same object in the image, their intensities are very similar. Therefore, the edge connecting these two pixels should be assigned to a high weight. We will address more on this issue in Chapter 3. The rest of this section will devote to the explanation of the general spectral clustering algorithm. Section 2.2.1 explain the formulation of the algorithm in the view of graph partitioning. Section 2.2.2 presents the algorithms.

2.2.1 Spectral Graph Partitioning

The idea of spectral clustering originates from spectral graph partitioning [20]. Spectral graph partitioning partitions a graph based on the similarity of the nodes. Intuitively, we would like to find a way to cut the graph into different parts such that nodes in the same part are similar, and nodes in different parts are dissimilar. In data analysis, each data point is considered as a node of the graph. By cutting the graph in this way, similar data points can be in the same cluster, and dissimilar data points can be in the different clusters.

Similarity Graph

Let $G = (V, E)$ be a weighted undirected graph with vertex set $V = \{v_1, ..., v_m\}$ and edge set $E = \{\{v_i, v_j\} |$ vertex i and vertex j are connected$\}$. The weight corresponding to

$\{v_i, v_j\}$ is w_{ij}. Since the graph is undirected, the weight $w_{ij} = w_{ji}$. The affinity weight matrix is defined to be a symmetric matrix W, where $W_{ij} = w_{ij}$. Define $d_i = \sum_{j=1}^{m} w_{i,j}$ to be the weighted degree of the vertex v_i, and the degree matrix D to be a diagonal matrix with diagonal $\{d_1,...,d_m\}$.

Consider any two sets of vertices A and B. Define a metric to measure the similarity between cluster A and cluster B:

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}. \tag{2.2}$$

The function $W(A, B)$ simply sums up the weights of the edges connecting cluster A and cluster B, which corresponds to the inter-region similarity in the image. Define a metric to measure the similarity between pixels in cluster A:

$$vol(A) = \sum_{i \in A} d_i. \tag{2.3}$$

The function $vol(A)$ simply sums up the weighted degree of each vertex in cluster A, which corresponds to the intra-region similarity in the image.

There are three different constructions of similarity graphs. They include: the ϵ-neighbourhood graph, the k-nearest neighbour graph, and the fully connected graph. In the the ϵ-neighbourhood graph, two points are connected only if their pairwise distance is within the parameter ϵ. In the k-nearest neighbour graph, a point v is connected to a point w only if v is one of the k-nearest neighbours of w. Finally, the fully connected graph is a complete graph with all the edge weights being nonzero.

Approximating graph-cut

Given any two sets of vertices A and B, the graph-cut of these two sets is defined to be the following:

$$cut(A, B) = W(A, B) = \sum_{i \in A, j \in B} w_{ij}. \tag{2.4}$$

To separate these two sets with minimal linkage broken, we can find a partition such that (2.2) is minimized. However, in most of the cases, the minimum cut separates the these two sets by isolating the point with minimal edge weight and the rest of the graph, which is not we want. We prefer to cut the graph into two balanced sets instead of some isolated points

and a very large set. Some modified versions of graph-cuts were proposed to consider the sizes of clusters, namely the Ratio Cut:

$$RatioCut(A,B) = \frac{cut(A,B)}{|A|} + \frac{cut(A,B)}{|B|}, \tag{2.5}$$

and the Normalized Cut:

$$NCut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}. \tag{2.6}$$

Here, $|A|$ denotes the cardinary of the set A, and $vol(A)$ is defined in (2.3). They both can be considered as a measure of the cluster size of A. Minimizing (2.5) or (2.6) not only cut off the two clusters with minimal linkages broken, but also try to keep the two clusters balanced as measured by its cardinary or total edge weights respectively.

The problems of minimizing Ratio Cut and Normalized Cut are very similar, because they are only using different measures for the cluster sizes. We will give a detail solution method of minimizing (2.6), and just state the solution of minimizing (2.5) due to space limitation.

Minimizing the NCut

In order to minimize the inter-region similarity, and to maximize the intra-region similarity, we consider the following optimization problem:

$$\min_{V_1,..,V_k} \frac{1}{2}\sum_{i=1}^{k} \frac{W(V_i,\bar{V}_i)}{vol(V_i)}, \tag{2.7}$$

where V_i corresponds to the ith cluster in the data set. The objective value is small as long as the cluster V_i is not too small for any i. Actually, the quantity $\sum_{i=1}^{k} \frac{1}{vol(V_i)}$ is minimized when all $vol(V_i)$ coincide, and the partition is balanced.

Let k be the number of clusters we want to classify.

Case 1 $k = 2$**:** Given any subset $V_1 \subset V$, the indicator vector

$$\mathbf{x} = \begin{cases} \sqrt{\frac{vol(\bar{V}_1)}{vol(V_1)}}, & \text{if } v_i \in V_1 \\ -\sqrt{\frac{vol(V_1)}{vol(\bar{V}_1)}} & \text{if } v_i \in \bar{V}_1 \end{cases} \tag{2.8}$$

defines the classification of the nodes in the graph. Define the Laplacian matrix L to be $L = W - D$. In [20], the author has shown that $x^T Lx = vol(V)Ncut(V_1, \bar{V_1})$, $(Dx)^T \mathbf{1} = 0$, and $x^T Dx = vol(V)$. Then, the minimization problem (2.7) can be written as:

$$\begin{aligned} \underset{V_1}{\text{minimize}} \quad & x^T Lx \\ \text{subject to} \quad & x \text{ has the form defined in (2.8)}, \\ & (Dx)^T \mathbf{1} = 0, \\ & x^T Dx = vol(V). \end{aligned} \tag{2.9}$$

However, this problem is NP hard. Relax (2.9) by considering all $x \in \mathbb{R}^n$. Also, by substituting $y = D^{\frac{1}{2}} x$ and letting $L_{sym} = D^{\frac{-1}{2}} L D^{\frac{1}{2}}$, (2.9) becomes:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & y^T L_{sym} y \\ \text{subject to} \quad & y^T (D^{\frac{1}{2}} \mathbf{1}) = 0, \\ & y^T y = vol(V). \end{aligned} \tag{2.10}$$

Since $vol(V)$ is a constant, (2.10) is equivalent to the following problem:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \frac{y^T L_{sym} y}{y^T y} \\ \text{subject to} \quad & y^T (D^{\frac{1}{2}} \mathbf{1}) = 0. \end{aligned} \tag{2.11}$$

By Rayleigh-Ritz theorem, the solution to (2.11) is given by the second eigenvector of L_{sym}, which gives the objective value being the second eigenvalue of L_{sym}[1].

Case 2 $k \geq 3$**:** Let H be a $n \times k$ matrix whose nonzero entries of the jth column define the elements in the jth cluster. More specifically, we define the jth column of H to be $h_j = (h_{1,j}, ..., h_{n,j})^T$,where

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{vol(V_j)}}, & v_j \in V_j \\ 0, & \text{otherwise}, \end{cases} \tag{2.12}$$

[1]We assumed that all eigenpairs are sorted according to the eigenvalues in ascending order.

for $i = 1, .., n$ and $j = 1, ..., k$.

Similar to the previous case, we have $H^T H = I$, ${h_i}^T D h_i = 1$, and ${h_i}^T L h_i = cut(V_i, \bar{V}_i)/vol(V_i)$. The problem of minimizing the Normalized Cut can be written as:

$$\begin{aligned} \underset{V_1,...,V_k}{\text{minimize}} \quad & Tr(H^T L H) \\ \text{subject to} \quad & H \text{ has the form defined in (2.12)}, \\ & H^T D H = I. \end{aligned} \tag{2.13}$$

Relax H to be a matrix in $\mathbb{R}^{n \times k}$, and substitute $P = D^{\frac{1}{2}} H$, problem (2.13) becomes:

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & Tr(P^T L_{sym} P) \\ \text{subject to} \quad & P^T P = I. \end{aligned} \tag{2.14}$$

This is a trace minimization problem of a real symmetric matrix. The solution of (2.14) is given by $[v_1, ..., v_k]$, the first k eigenvectors of L_{sym}.

Minimizing the Ratio Cut

The procedure of minimizing the Ratio Cut is very similar to that of the Normalized Cut, so we will only outline the solutions here.

Problem:

$$\min_{V_1,..,V_k} \frac{1}{2} \sum_{i=1}^{k} \frac{W(V_i, \bar{V}_i)}{|V_i|},$$

where A_i corresponds to the ith cluster in the data set.

Case 1 $k = 2$: Define the indicator vector to be:

$$\mathbf{x} = \begin{cases} \sqrt{\frac{|\bar{V}_1|}{|V_1|}}, & \text{if } v_i \in V_1, \\ -\sqrt{\frac{|V_1|}{|\bar{V}_1|}}, & \text{if } v_i \in \bar{V}_1. \end{cases}$$

Similar to the steps in minimizing Normalized Cut, we obtain the following problem:

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \frac{x^T L x}{x^T x} \\ \text{subject to} \quad & x^T \mathbf{1} = 0. \end{aligned} \tag{2.15}$$

By Rayleigh-Ritz theorem, the solution to (2.15) is given by the second eigenvector of L.

Case 2 $k \geq 3$ Define the jth indicator vector for cluster j to be:

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{|V_j|}}, & v_j \in V_j \\ 0, & \text{otherwise}, \end{cases} \tag{2.16}$$

for i=1,..,n.

Let H be a $n \times k$ matrix such that $H_{i,j} = h_{i,j}$. Similar to the problem of minimizing a Normalized Cut, we have:

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & Tr(H^T L H) \\ \text{subject to} \quad & H^T H = I, \end{aligned} \tag{2.17}$$

whose solution is given by $[u_1, ..., u_k]$, the first k eigenvectors of L.

2.2.2 Spectral Clustering: The algorithm

The relaxization of minimizing the Ratio Cut leads to the unnormalized spectral clustering, and the relaxization of the Normalized Cut leads to the normalized spectral clustering. Algorithm 1 and Algorithm 2 below outline the steps of the unnormalized and normalized spectral clustering respectively.

Algorithm 1: Unnormalized Spectral Clustering

Input: A set of data points $\{x_1, ..., x_n\}$, a number k representing the number of clusters

Output: Clusters $V_1, ..., V_k$ with $V_i = \{i | x_i \in C_i\}$

1 **Construct a similarity graph of $\{x_1, ..., x_n\}$, and let W be the corresponding affinity weight matrix;**
2 **Compute $L = D - W$;**
3 **Compute the first k eigenvectors $v_1, ..., v_k$ of L;**
4 **Let U be a $n \times k$ matrix such that $U = [v_1, ..., v_k]$;**
5 **Consider each row of U as a data point and cluster these points using the k-means algorithm into clusters $C_1, ..., C_k$;**

Algorithm 2: Normalized Spectral Clustering

Input: A set of data points $\{x_1, ..., x_n\}$, a number k representing the number of clusters

Output: Clusters $V_1, ..., V_k$ with $V_i = \{i | x_i \in C_i\}$

1 **Construct a similarity graph of $\{x_1, ..., x_n\}$, and let W be the corresponding affinity weight matrix;**
2 **Compute $L = D - W$ and $L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$;**
3 **Compute the first k eigenvectors $v_1, ..., v_k$ of L_{sym};**
4 **Let U be a $n \times k$ matrix such that $U = [v_1, ..., v_k]$;**
5 **Consider each row of U as a data point and cluster these points using the k-means algorithm into clusters $C_1, ..., C_k$;**

2.3 Robust Principal Component Analysis for Background Subtraction

Background Subtraction describes the process of isolating moving objects from the static background in a video surveillance. Many background subtraction methods have been developed in the past few years, which include the non-parametric density models [7], the Eigenbackgrounds approach [16], and the Robust Principal Component Analysis (RPCA) [11]. The RPCA approaches are adopted in most applications because of its efficiency and effectiveness. In this section, we will review the technique of RPCA via Principal Component Pursuit (PCP) [3], and briefly mention how it works in moving foreground

detection. In Chapter 3, several cell segmentation methods based on these ideas will be proposed. Before the discussion of RPCA, we will give a brief introduction on Alternating Direction Method of Multipliers (ADMM) as this is the method to solve the optimization problem associated to PCP.

2.3.1 Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) are based on the method of multipliers [12]. The method of multipliers is a robust modification of the dual ascent method, which provides convergence for not strictly convex functions. Consider the following problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & Ax = b. \end{aligned} \tag{2.18}$$

The augmented Lagrangian for (2.18) is $\mathbf{L} = f(x) + y^T(Ax - b) + (\frac{\rho}{2})\|Ax - b\|_2^2$, where y is the dual variable, and $\rho > 0$ is the penalty parameter. The augmented Lagrangian is the standard Lagrangian of the following problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) + (\frac{\rho}{2})\|Ax - b\|_2^2 \\ \text{subject to} \quad & Ax = b. \end{aligned} \tag{2.19}$$

The minimization problems (2.18) and (2.19) are equivalent since any feasible solutions would give the same objective function in both problems. The purpose of adding the ℓ_2 norm term is to avoid the ill-conditioning of the problem when f is not strictly convex. The extra ℓ_2 norm term brings two issues. Firstly, the primal feasibility is not obviously imposed in each iterative update. However, the extra ℓ_2 norm penalization term acts as a price incurred by infeasibility. In fact, it can be shown that the primal residual $Ax^{k+1} - b$ converges to zero eventually. Secondly, the extra ℓ_2 term ruins the decomposability of the objective function even when f is separable. Consider the following problem:

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & f_1(x) + f_2(z) + (\frac{\rho}{2})\|Ax + Bz - b\|_2^2 \\ \text{subject to} \quad & Ax + Bz = b. \end{aligned} \tag{2.20}$$

It would be more efficient if we could split (2.20) for updating the primal in each interation. ADMM addresses this problem by minimizing the augmented Lagrangian $\mathbf{L}$ over one

variable at a time while keeping all others fixed. Below outlines the steps of ADMM for solving (2.20):

$$x^{k+1} := \underset{x}{\text{argmin}} \quad \mathbf{L}(x, z^k, y^k), \tag{2.21}$$

$$z^{k+1} := \underset{z}{\text{argmin}} \quad \mathbf{L}(x^{k+1}, z, y^k), \tag{2.22}$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \tag{2.23}$$

It can be shown that if the functions f_1 and f_2 satisfy the following two assumptions:

- The functions f_1 and f_2 are closed, proper, and convex[2].
- The regular Lagrangian of (2.20) has a saddle point.

Then the iteration of ADMM satisfies the following three types of convergences:

- The residual $r^k = Ax^k + Bz^k - c$ goes to zero when k is sufficiently large.
- The objective function $f_1(x^k) + f_2(z^k)$ approaches the optimal value when k is sufficiently large.
- The dual variable y^k approaches the dual optimal point when k is sufficiently large.

The proof of above result can be found in [1]. The solution of ADMM converges to true solution with moderate accuracy reasonably fast in most applications [1], but it takes many iterations for it to attain a high accuracy. Therefore, ADMM is useful in most statistical and machine learning problems, where moderate accurate solutions are sufficiently to give good predictions.

2.3.2 RPCA via PCP

Given an $m \times n$ data matrix M. The objective is to find a decomposition of M such that $M = L_0 + S_0$, where L_0 is low-rank and S_0 is sparse. Intuitively, we can find the

[2]A function is closed, proper, and convex if and only if its epigraph is a closed nonempty convex set.

decomposition by solving the following optimization problem:

$$\begin{aligned} \underset{L,S}{\text{minimize}} \quad & Rank(L) + \lambda \|S\|_0 \\ \text{subject to} \quad & L + S = M, \end{aligned} \tag{2.24}$$

where λ is a parameter to control the sparsity of S. This problem looks very simple, but actually it is NP-hard. In fact, we need to perform a search in a restricted solution space, which has combinatorial complexity [11]. Candes et al. [3] proposed relaxing (2.24) by the Principal Component Pursuit (PCP):

$$\begin{aligned} \underset{L,S}{\text{minimize}} \quad & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} \quad & L + S = M, \end{aligned} \tag{2.25}$$

where $\|L\|_* := \sum_i \sigma_i(L)$[3] denotes the nuclear norm of L, and $\|S\|_1 = \sum_{i,j} |S_{ij}|$ denotes the vector-version ℓ_1 norm. Candes et al. [3] has shown that if the problem is well-posed[4], then the decomposition can be recovered exactly with very high probability, provided that the rank of the low-rank component L and the ℓ_0 norm of the sparse component S satisfies the following conditions:

$$rank(L) \leq \frac{\rho_r \max(m,n)}{\mu \log(\min(m,n))^2}, \quad \|S\|_0 \leq \rho_s mn, \tag{2.26}$$

where ρ_r and ρ_s are some positive constants.

PCP can be solved using the alternating direction method of multipliers (ADMM) [3]. The augmented Lagrangian is:

$$\mathbb{L}(L, S, Y) = \|L\|_* + \lambda \|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2} \|M - L - S\|_F^2. \tag{2.27}$$

As mentioned in Section 2.3.1, ADMM suggests recursively updating $L, S, and\ Y$ according to the following:

$$L^{k+1} := \underset{L}{\text{argmin}} \quad \mathbb{L}(L, S^k, Y^k), \tag{2.28}$$

[3] $\sigma_i(L)$ is the ith singular value of L

[4] L_0 and S_0 satisfy some conditions [3], which implies that L_0 and S_0 cannot be sparse and low-rank at the same time.

$$S^{k+1} := \underset{S}{\text{argmin}} \quad \mathbb{L}(L^{k+1}, S, Y^k), \tag{2.29}$$

$$Y^{k+1} := Y^k + \mu(L^{k+1} + S^{k+1} - M). \tag{2.30}$$

It can be shown that[5]:

$$(\underset{S}{\text{argmin}} \quad \mathbb{L}(L, S, Y)) = \mathcal{S}_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y), \tag{2.31}$$

$$(\underset{L}{\text{argmin}} \quad \mathbb{L}(L, S, Y)) = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y), \tag{2.32}$$

where $\mathcal{S}$ and $\mathcal{D}$ are the shrinkage operators defined below:

$$\mathcal{S}_\tau(x) = sign(x)\max(|x| - \tau, 0), \quad \mathcal{D}_\tau(X) = U\mathcal{S}_\tau(\Sigma)V^T{}^{6}.$$

The order of updating (2.28) and (2.29) are not crucial. However, the final results might be different with different updating order, because the dual update (2.30) is then done after the update of L and before the update of S. Algorithm 3 outlines the steps of solving PCP with ADMM.

Algorithm 3: PCP by ADMM

Input: $m \times n$ data matrix M,parameter μ
Output: $m \times n$ matrix L and $m \times n$ matrix S

initialize $S^0 = L^0 = 0_{m\times n}$;

while not converge do:
  compute $L^{k+1} = \mathcal{D}_{\mu^{-1}}(M - S^k + \mu^{-1}Y^k)$;
  compute $S^{k+1} = \mathcal{S}_{\lambda\mu^{-1}}(M - L^{k+1} + \mu^{-1}Y^k)$;
  compute $Y^{k+1} = Y^k + \mu(M - L^{k+1} - S^{k+1})$;

end while
output S **and** L.

[5]The derivations will be shown in Appendix A.
[6]$X = U\Sigma V^T$ is the singular value decomposition of X.

Chapter 3

Methodology

Cell image segmentation describes the process of decomposing a microscopic image into cells and background. This problem is different from other image segmentation problems in computer vision. The major challenge comes from the low variation of pixel intensities between cells and background. Classical segmentation techniques introduced in Section 2.1 cannot give a good segmentation results. Therefore, some state-of-the-art methods are required to tackle this problem. There are two ways to think about this problem. It can be casted as a large-scale clustering problem, which comes down to classifying pixels into cell pixels and background pixels. It can also be considered as a background subtraction problem, where the cells are the moving objects we want to isolate from the static background.

Spectral clustering is one of the most popular clustering algorithms in machine learning. It is good at clustering data with implicit connection. Intuitively, it maps the data to user-defined feature space, and then performs k-means clustering in the low-dimensional subspace spanned by the first k eigenvectors of the corresponding Laplacian matrix. This makes spectral clustering particularly fit the cell segmentation problem. Because of the low cell-to-background contrast, if we classify the pixels directly with k-mean clustering based on the pixel intensity, the algorithm will wrongly classify all pixels into a single cluster. To overcome this problem, we incorporate the physical distance of pixels into the similarity measurement, and perform spectral clustering using the defined similarity matrix. A hierarchical spectral clustering approach to capture cell boundary automatically is proposed in Section 3.1.1. We have also made two modifications to the hierarchical approach to better fit the problem. The variations will be presented in Section 3.1.2 and Section 3.1.3 respectively.

If a sequence of consecutive cell images is given, we can consider the segmentation problem as identifying the moving cells over time. We use Robust Principal Component Analysis (RPCA) introduced in Section 2.2 to locate the cells. However, there are issues when using RPCA in our problem setting. Since the data we deal with are images, it is reasonable to assume that their decompositions having non-negative entries as well. Therefore, We propose to add non-negative constraints to the Principal Component Pursuit (PCP) to ensure non-negativity of component entries. Also, we have used a different approximation of the zero-norm to suit our problem better. These methods will be introduced in Section 3.2.

The segmented regions from the spectral clustering methods are spatially consistent, forming a closed connected set of pixels. The background in the segmented result usually does not have any misclassified cell pixels in it. However, this often results in under-segmentation of the cells because some cell pixels are considered to be very similar to the background in the affinity weight matrix. The RPCA method, on the other hand, which exploits information of other frames, can capture more close-to-background cell pixels. However, the intensities are not exactly the same in the background for different frames due to artifacts, which give rises to scattered noises in the background in the RPCA methods. In addition, the segmented regions in the RPCA methods are sometime not a connected set of pixels. Our objective is to improve their deficiencies by combining the two techniques.

It is not totally obvious how to combine the two methods since the spectral clustering methods and the RPCA methods are very different. They are developed from two very different points of view: one is considered as a clustering problem, and the other is considered as a static background-modelling problem. A straight forward way to combine these two methods is to formulate it as a two-stage approach. Two formulations of the stepwise methods are proposed in Section 3.3.1. They combine spectral clustering with RPCA step-by-step. The stepwise methods do not improve the segmentation results significantly, because the two methods act independently. Ideally, there should be interaction between the two methods in the combined method so that the combined method is superior to any individual method.

A more sophisticated way to combine the spectral clustering with RPCA is to formulate the problem as a Principal Component Pursuit (PCP) with graph-cut penalization. We add an extra penalty to PCP to measure how well the partition is with respect to its graph-cut. The problem comes down to making the connection between PCP and graph-cut in the optimization. We propose two ways for the connection, and give the derivations of the two methods in Section 3.3.2.

3.1 Hierarchical spectral clustering methods for image segmentation

In Chapter 2, we have mentioned that spectral clustering can be used in image segmentation by constructing a similarity matrix to reflect the structure of the image. However, in bright-field microscopy, cell pixels have very similar intensity values as the background pixels. It is likely that the similarity measure of two background pixel intensities is almost the same as the one of a cell pixel and a background pixel. Therefore, we need to consider additional local grouping cues. The physical distance between two pixels is simple to measure. In particular, let the weight $w_{i,j}$ of each edge be a similarity function between pixel i and j in the given image. We can use the Gaussian function to measure the similarity between each pair of pixels:

$$w_{i,j} = e^{\frac{-|I(x_i,y_i)-I(x_j,y_j)|^2}{\sigma_I}} e^{\frac{-|(x_i,y_i)-(x_j,y_j)|_2^2}{\sigma_d}},$$

where $I(x, y)$ is a function that computes the intensity value of a pixel at location (x, y), and σ_I and σ_d are the input parameters, which penalize on pixel intensities and spatial distances of pixels respectively. Therefore, σ_I and σ_d need to be tuned to obtain the optimal results depending on the nature of the graph. In order to be able to create disjoint clusters, we construct a graph that has local connection only. In Chapter 2, three constructions of similarity graphs are presented. For large scale clustering problem like image segmentation, the $\epsilon - neighborhood$ graph is most appropriate in terms of efficiency as it will produce a sparse banded Laplacian matrix. In our implementation, we construct the similarity graph using the $\epsilon - neighborhood$ method.

3.1.1 Image segmentation as a spectral clustering problem

Spectral clustering can be applied directly to the cell images, but it is hard to determine the number of clusters in the algorithm. The ideal case is to choose two clusters, so one cluster represents the foreground and the other represents the background. However, since the intensity values in the cell images are very close, pixels are not different enough to give a good result of the two-class clustering. Choosing two clusters will often give an under-segmentation result with noises in the background and holes inside the cells. On the other hand, choosing too many clusters will result in a multiple clusters of a single cell. However, the clusters comprised the cell can capture the cell body precisely such that it can even match up the curvature of the cell boundaries. To be able to capture the cells without human intervention, we utilize both the two-classes clustering result and the

k-classes clustering result, where k is a large integer. The two-classes clustering result is used to detect the cell location, and can be used to determine which of the k clusters are cells. More specifically, if the majority of the points in a cluster C of the k-classes result are the points of foreground in the two-classes result, then C is classified as cells. We use the coarse two-classes segmentation result as a mask to identify the labels of each super-pixel obtained from the k-classes spectral clustering. The algorithm of this method is stated below.

Algorithm 4: Spectral clustering for image segmentation

Input: Image M, parameter $\sigma_i, \sigma_{xy}, \varrho_i, \varrho_{xy}$, number of clusters k, $n \times n$ zero matrix $\bar{M}$

Output: Binary matrix $\bar{M}$

1 **Construct the weighted affinity matrix W_1 of M using the RBF kernel with bandwidth σ_i and σ_{xy}:**

$$w_{i,j} = e^{\frac{-|I(x_i,y_i)-I(x_j,y_j)|^2}{\sigma_I}} e^{\frac{-|(x_i,y_i)-(x_j,y_j)|_2^2}{\sigma_d}},$$

and compute the Laplacian matrix using $L_1 = Diag(sum(W_1)) - W_1$;

2 **Solve for the eigenvector v corresponding to the second smallest eigenvalue of L_1;**

3 **Classify the pixels of M based on the sign of v, and obtain the corresponding binary matrix M^* with cell pixels labelled by ones;**

4 **Construct the Laplacian matrix L_2 as step 1 using bandwidth ϱ_i and ϱ_{xy} respectively;**

5 **Solve for the k+1 eigenvectors that corresponding to the k+1 smallest eigenvalues of L_2;**

6 **Disregard the eigenvector corresponding to the smallest eigenvalue, and form a $n^2 \times k$ matrix U using the remaining eigenvectors;**

7 **Run k-means to cluster the rows of U into k segments;**

8 **Use M^* obtained from step 3 to classify the m segments obtained from step 7 into cell segments and background segments;**

9 **Set all pixels in $\bar{M}$ corresponding to cell segments to ones;**

3.1.2 Spectral clustering on an image volume

One very important fact of the cell image segmentation problem is that we are given a sequence of cell images taken over different time instead of a single image. We might assume that cells do not move dramatically over frames. This assumption is reasonable as one can control the number of pictures to be taken over a unit of time to ensure small movements of cells. In this case, the shape and the location of cells at earlier time can help the segmentation of cells at current time. To realize this idea in the spectral clustering approach, we consider a classification of voxels instead of pixels. To be specific, we align the sequence of images together as a volume, and apply the previous method on the 3D image volume. Similar to Method 1, we use the Gaussian kernel function to measure the similarity between each pair of voxels:

$$w_{i,j} = e^{\frac{-|I(x_i,y_i,z_i)-I(x_j,y_j,z_j)|^2}{\sigma_I}} e^{\frac{-|(x_i,y_i,z_i)-(x_j,y_j,z_j)|_2^2}{\sigma_d}},$$

where the z-coordinate corresponds to the layer of the image volume. The Laplacian matrix is then compute using $L = Diag(sum(W)) - W$[1]. This method will improve the segmentation result because it considers additional information. The new Laplacian matrix reflects the relationship between pixels at the current time frame and the ones at the consecutive time frames. In particular, if a pixel at time frame t is a pixel of cell, our method will put more weights on the label of cell for that pixel at time frame $t + 1$. However, the number of rows (columns) of the square Laplacian matrix equals to the number of voxels in the image volume. The computation is more expensive, because we need to solve an eigen-decomposition problem for a much larger matrix. The detail of this method is outlined in Algorithm 5.

3.1.3 Recursive spectral clustering

Another difficulty of segmenting the cells is that the intensity contrast is not uniform between cells or even within a single cell. Using the previous methods may give under-segmentation results. The parts of the cells with higher intensity contrast can be segmented out, but the parts with lower contrast are usually identified as background mistakenly. To overcome this problem, one small variation can be made to improve the segmentation result. The idea is to perform the previous method recursively. We first obtain an initial under-segmentation result by using one of the previous methods, and then perform

[1] $sum(W) = \sum_j W(:, j)$

Algorithm 5: Spectral clustering for 3D image segmentation

Input: a sequence of $n \times n$ images M_1,..., M_k, parameter $\sigma_i, \sigma_{xy}, \varrho_i, \varrho_{xy}$, and number of clusters m, $n \times n \times m$ zero matrix $\bar{M}$

Output: A three dimensional matrix of binary numbers

1 **Stack M_1,...,M_k together to form a three-dimensional matrix M such that the sth page of M corresponds to M_s;**

2 **Construct the weighted affinity matrix W_1 of M using the RBF kernel with bandwidth σ_i and σ_{xy}:**

$$w_{i,j} = e^{\frac{-|I(x_i,y_i,z_i)-I(x_j,y_j,z_j)|^2}{\sigma_I}} e^{\frac{-|(x_i,y_i,z_i)-(x_j,y_j,z_j)|_2^2}{\sigma_d}},$$

and compute the Laplacian matrix using $L_1 = Diag(sum(W_1)) - W_1$;

3 **Solve for the eigenvector v corresponding to the second smallest eigenvalue of L_1;**

4 **Classify the voxels of M based on the sign of v and obtain the corresponding binary matrix M^* with cell voxels labelled by ones.**

5 **Construct the Laplacian matrix L_2 as step 1 using bandwidth ϱ_i and ϱ_{xy} respectively;**

6 **Solve for the k+1 eigenvectors that corresponding to the k+1 smallest eigenvalues of L_2;**

7 **Disregard the eigenvector corresponding to the smallest eigenvalue, and form a $(k \times n^2) \times m$ matrix U using the remaining eigenvectors;**

8 **Run k-means to cluster the rows of U into k segments;**

9 **Use M^* obtained from step 3 to classify the m segments obtained from step 8 into cell segments and background segments;**

10 **Set all voxels in $\bar{M}$ corresponding to cell segments to ones;**

spectral clustering again on the initial result. To do this, we mark all pixels classified as foreground previously to be one class by assigning the value 1 to corresponding entries in the normalized weight matrix W. We fill in other values of W using the Gaussian similarity function as before, and perform the k-classes clustering. The method is summarized in the following:

Algorithm 6: Recursive spectral clustering for image segementation

Input: $n \times n$ images M, $n \times n$ binary image $\tilde{M}$, parameter $\sigma_i, \sigma_{xy}, \varrho_i, \varrho_{xy}$, number of clusters m, $n \times n$ zero matrix $\bar{M}$

Output: Binary Image $\bar{M}$

1 **Construct the weighted affinity matrix W_1 of M using the RBF kernel with bandwidth σ_i and σ_{xy}:**

$$w_{i,j} = e^{\frac{-|I(x_i,y_i)-I(x_j,y_j)|^2}{\sigma_I}} e^{\frac{-|(x_i,y_i)-(x_j,y_j)|_2^2}{\sigma_d}},$$

and compute the Laplacian matrix using $L_1 = Diag(sum(W_1)) - W_1$;

2 **Solve for the eigenvector v corresponding to the second smallest eigenvalue of L_1;**

3 **Classify the pixels of M based on the sign of v, and obtain the corresponding binary matrix M^* with cell pixels labelled by ones;**

do:

4 **Construct the weight similarity matrix using bandwidth ϱ_i and ϱ_{xy} respectively;**

5 **For every pixel p labelled as foreground in $\tilde{M}$, set $W(p,q) = 1$ if pixel q is labelled as foreground in $\tilde{M}$;**

6 **Construct the Laplacian matrix L_2 associate to W;**

7 **Solve for the m+1 eigenvectors that corresponding to the m+1 smallest eigenvalues of L_2;**

8 **Disregard the eigenvector corresponding to the smallest eigenvalue, and form a $n^2 \times m$ matrix U using the remaining eigenvectors;**

10 **Run k-means to cluster the rows of U into m segments;**

until converge

11 **Use M^* obtained from step 3 to classify the m segments obtained from step 10 into cell segments and background segments;**

12 **Set all pixels in $\bar{M}$ corresponding to cell segments to ones;**

3.2 Image Segmentation as a sparsity pursuit problem

Robust principal component analysis (RPCA) is a well-received method in data analysis. Its principal objective is to reduce the dimension of a data set to aid higher level processing, which coincides with the objective of image segmentation. However, RPCA has rarely been used on image segmentation, because in most cases only a single image is given to segmentation. For cell image segmentation, a consecutive sequence of cell images is given. These images usually record the cell activities, where cells move slightly over consecutive frames. Thus, it is intuitive to use RPCA to aid the segmentation since RPCA can be used to capture the moving objects in the static background.

3.2.1 Detection of moving cells

RPCA has been successful in the detection of foreground from the static background in a video surveillance system. RPCA gives an accurate approximation to the probability distribution of the background and foreground by decomposing a data matrix into a low-rank component L and a sparse component S, where L recovers the background and S recovers the moving foreground. If we consider each cell image as a frame in a video, we can use RPCA to capture the moving cells. First, a data matrix M can be formed by appending each cell image as the column of M. We decompose the data matrix M by solving the related sparsity pursuit problem. The sparsity pursuit problem is formulated as a convex optimization problem to minimize the sum of the nuclear norm of the low-rank component L and the ℓ_1 norm of the sparse component S subject to a linear constraint:

$$\begin{aligned} \underset{L,S}{\text{minimize}} \quad & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} \quad & L + S = M. \end{aligned}$$

The problem can be solved using the alternating direction method of multipliers (ADMM) whose solution converges to the true solution with moderate accuracy reasonably fast [1]. After we have obtained the sparse component S, we consider each column of S as an indicator vector with zero labeled background and non-zero labeled cells. However, while S is a sparse matrix, it may have nonzero entries distributed everywhere. In fact we have observed that there are many zero entries inside the cell body, and many nonzero entries in the background. These are the misclassified points in our case. Some post processing

is needed to clean up the results, and to remove the scattered noise in the background. In particular, we first remove all small clusters with less than 10 pixels[2], and then fill the holes inside the cells with morphological operations. Finally, a standard curve evolution method [4] is applied on the binary image to capture the cell boundaries. We summarize the algorithm below.

Algorithm 7: Robust Principal Component Analysis for image segmentation

Input: a sequence of images M_1,..., M_n, parameter μ, λ
Output: a sequence of binary images S_j, $j = 1, .., n$

Form a data matrix M, where each column of M corresponds to a cell image;
initialize S,L,Y;

while not converge do:
  compute $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$;
  compute $S = \mathcal{S}_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y)$;
  compute $Y = Y + \mu(M - L - S)$
end while
reshape each column of S to be the size of the original image remove small clusters in each S_j;
fill holes inside S_j **using morphological operations;**
perform Chan-Vese active contour method on each S_j;

output S_j $j = 1, ..., n$

3.2.2 PCP with non-negative constraints

If we consider each image as a topographical map with the intensity value of a pixel interpreted as its altitude on the map, then applying RPCA is like slicing the map into two parts: one part contains the cells and the other part contains the background. Since the background part is forced to be identical for every image in the sequence, negative entries might be in the components from the decomposition. For images with inconsistent pixel intensity in the background, RPCA gives scattered negative entries in the background.

[2]This can be done using the MATLAB built in function: bwareaopen.

Therefore, we can enforce non-negative constraints to reduce the number of missed classified points in the sparse component S:

$$\begin{aligned} \underset{L,S}{\text{minimize}} \quad & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} \quad & L + S = M, \\ & L \geq 0,\ S \geq 0. \end{aligned}$$

The non-negativity constraints of L and S complicate the optimization problem. We can introduce auxiliary variables S+ and L+, and enforce non-negativity on these two variable, so the objective function does not involve variables with non-negativity constraints. More specifically, the non-negative sparsity pursuit problem can be rewritten in the following form:

$$\begin{aligned} \underset{L,S}{\text{minimize}} \quad & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} \quad & L + S = M, \\ & L = L+,\ S = S+, \\ & L+ \geq 0,\ S+ \geq 0. \end{aligned}$$

This problem can be solved by ADMM. We have added 4 more variables: two more primal $(L+, S+)$ and two more dual (Z, Q) to the augmented Lagrangian:

$$\begin{aligned} & \mathcal{L}(L, S, L+, S+, Y, Z, Q) = \\ & \|L\|_* + \lambda \|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2} \|M - L - S\|_F^2 + \\ & \langle Z, L - L+ \rangle + \langle Q, S - S+ \rangle + \frac{\gamma}{2} \|L - L + \|_F^2 + \frac{\gamma}{2} \|S - S + \|_F^2. \end{aligned} \tag{3.1}$$

Minimizing the augmented Lagrangian with respect to one variable at a time can solve the problem. Since the non-negativity constraints are applied to $L+$ and $S+$ only. We can use the same approaches as the ones described in Appendix A.1 and Appendix A.2 to derive the update formulas for L and S. For the update of $L+$ and $S+$, we consider the following problems:

$$\begin{aligned} \underset{L+}{\text{minimize}} \quad & \langle Z, L - L+ \rangle + \frac{\gamma}{2} \|L - L + \|_F^2 \\ \text{subject to} \quad & L+ \geq 0, \end{aligned}$$

and

$$\begin{aligned} \underset{S+}{\text{minimize}} \quad & \langle Q, S - S+ \rangle + \frac{\gamma}{2} \|S - S+\|_F^2 \\ \text{subject to} \quad & S+ \geq 0. \end{aligned}$$

The above two minimization problems are non-negative quadratic programmings, which have been well-studied in the literature. One simple approximation is to enforce the non-negativity constraints by projecting the usual update[3] onto the convex feasible set corresponding to the space defined by all the non-negative constraints [18]. The update formulas for L+ and S+ are: $L+ = max(L+\gamma^{-1}Z, 0)$ and $S+ = max(S+\gamma^{-1}Z, 0)$, respectively. Finally, we update the dual variables according to gradient ascent. The steps of this method are outlined in Algorithm 8.

3.2.3 Sparsity Pursuit with different ℓ_0 approximation

In the sparsity pursuit problem, ℓ_1 norm is used as an approximation of the ℓ_0 norm such that the objective function is convex and easier to solve. However, in our problem setting, the ℓ_1 penalty gives a solution S with small magnitude noises in the background, because minimizing the ℓ_1 of S usually gives a solution with moderate number of moderated-sized nonzero entries [19]. In our problem, we want S to be sparse with large magnitude nonzero entries. Thus, we replace the ℓ_1 penalty by another ℓ_0 norm approximation to force the entries in S to be either 0 or nonzero with relatively large magnitude. In particular, we use the approximation proposed by Zhang [9]:

$$g(S_{ij}) = \begin{cases} \nu|S_{ij}|, & \text{if } |S_{ij}| < \eta, \\ \nu\eta, & \text{otherwise,} \end{cases}$$

where ν and η are some predefined constant to control the magnitude of the nonzero entries.

The Zhangs penalty is a two-stage rescaled ℓ_1 penalty. The penalty is exactly ℓ_1 penalty for parameters with small magnitude. The parameters with relatively large magnitude are not penalized anymore. Figure. 3.1 shows the comparison between the ℓ_1 penalty and the Zhangs penalty.

[3]Since the objective function is strictly convex, the usual update is to enforce the first necessary condition

Algorithm 8: Non-negative Robust Principal Component Analysis for image segmentation

Input: a sequence of images M_1,..., M_n, parameter μ, λ, γ

Output: a sequence of binary images S_j, $j = 1...n$

Form a data matrix M, where each column of M corresponds to a cell image;

initialize S,L,$S+$,$L+$,Y,Z,Q;

while not converge do:

compute $L = \mathcal{D}_{(\mu+\gamma)^{-1}}(\mu(\mu+\gamma)^{-1}M - \mu(\mu+\gamma)^{-1}L + (\mu+\gamma)^{-1}Y + \gamma(\mu+\gamma)^{-1}(L+) - (\mu+\gamma)^{-1}Z)$;

compute $S = \mathcal{S}_{\lambda(\mu+\gamma)^{-1}}(\mu(\mu+\gamma)^{-1}M - \mu(\mu+\gamma)^{-1}L + (\mu+\gamma)^{-1}Y + \gamma(\mu+\gamma)^{-1}(S+) - (\mu+\gamma)^{-1}Q)$;

compute $L+ = max(L + \gamma^{-1}Z, 0)$;

compute $S+ = max(S + \gamma^{-1}Z, 0)$;

compute $Y = Y + \mu(M - L - S)$;

compute $Z = Z + \gamma(L - L+)$;

compute $Q = Q + \gamma(Q - Q+)$;

end while

reshape each column of S to be the size of the original image remove small clusters in each S_j;

fill holes inside S_j **using morphological operations;**

perform Chan-Vese active contour method on each S_j;

output S_j $j = 1, ..., n$

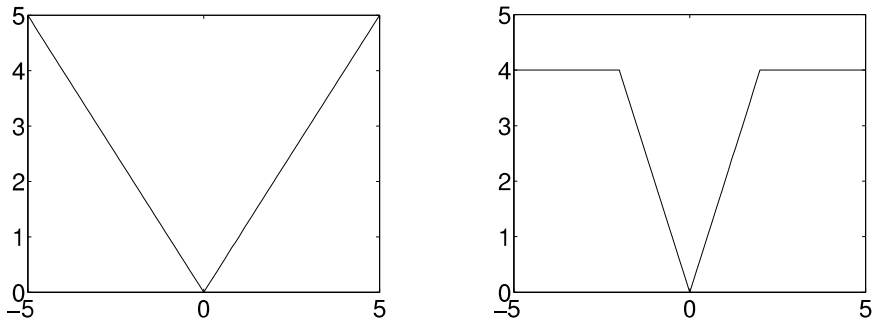


Figure 3.1: Comparison of ℓ_1 penalty (left) and the Zhang's penalty (right).

Since the function g is step function whose function value depends on the value of S_{ij}, we update S_{ij}^k, the value of S_{ij} at the kth iteration based on S_{ij}^{k-1}, the value of S_{ij} at the $k-1$ iteration. The detail is outlined in step 4 of Algorithm 9. Here, we do not enforce the first necessary condition of S exactly, but the numerical result in Chapter 4 suggests that this is a reasonable approximation. The steps of the RPCA with the modified approximated ℓ_0 penalty are outlined in Algorithm 9. This model is able to eliminate some noises in the background. However, one small detail need to be mentioned is that the parameter λ need to be smaller in this new model. Otherwise, some entries corresponding to the cell bodies will be forced to zero in this model.

3.3 Combinations of spectral clustering and RPCA

We have proposed three variations of image segmentation methods using spectral clustering, and three using RPCA in the previous two subsections. The spectral clustering methods produce nearly noiseless results. Classified pixels tend to get together, because the physical distance is incorporated into the computation of the affinity weight matrix. However, many foreground pixels are misclassified as background as their intensity values are identical to the ones of the background. Therefore, the methods based on spectral clustering usually gives under-segmentation results. Another two important drawbacks are the significant computation time and the sensitivity to tuning parameters. On the other hand, the RPCA based methods are more robust and efficient. They are also capable of capturing the closed-to-background cell pixels, because they link the cell segments at different time frame. However, the major drawback of the RPCA approach is that they produce results with scattered noises in the background. Also, some entries in the cells are set to be zero, so the segmented cell regions are not closed continuous sets, which causes difficulty in the post-processing step. In this section, we will present two novel methods linking RPCA and spectral clustering as the principal component pursuit with graph-cut penalization. The two methods are in the hope to offset the weaknesses of the previous proposed methods. Before we present the algorithms, we will show two methods that combine spectral clustering and RPCA as two separate steps. These two methods cannot give great improvements to the segmentation results, but they are worth mentioning because they might provide insights for future research. The two stepwise methods will be presented in Section 3.3.1, and the two novels methods will be presented in Section 3.3.2.

Algorithm 9: Robust Principal Component Analysis for image segmentation with Zhang's penalty

Input: a sequence of images M_1,..., M_n, parameter μ, λ, η
Output: a sequence of binary images S_j, $j = 1, .., n$

Form a data matrix M, where each column of M corresponds to a cell image;
initialize S,L,Y;

while not converge do:
  compute $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$;

  if $|S_{ij}| \leq \eta$: **compute** $S = \mathcal{S}_{\lambda\mu^{-1}}(M_{ij} - L_{ij} + \mu^{-1}Y_{ij})$
  else: **compute** $S_{ij} = M_{ij} - L_{ij} + \mu^{-1}Y_{ij}$;
  compute $Y = Y + \mu(M - L - S)$;

end while
reshape each column of S to be the size of the original image remove small clusters in each S_j;
fill holes inside S_j **using morphological operations;**
perform Chan-Vese active contour method on each S_j;

output S_j $j = 1, ..., n$

3.3.1 Two-steps Methods

The first method is very similar to the hierarchical spectral clustering method in Section 3.1. Instead of using spectral clustering twice, we can use RPCA to obtain a coarse segmentation to separate cells and background. Similar to the methods in Section 3.1, we use the two-class result as a mask to classify the k segments from the k-classes spectral clustering into cell segments and background segments, and thus obtain a clearer segmentation. This method produces very similar results as the hierarchical spectral clustering method, because the two classes clustering result is not a deterministic step in the method. It only uses as a mask to help identifying cell locations. However, this method performs better than hierarchical spectral clustering in term of efficiency and robustness. RPCA performs much faster, and their result is less dependent on the parameters in the model. The steps are outlined in Algorithm 10.

The second method is formulated as a normalized cut with partial grouping constraints [24]. It proceeds in two steps: in the first step, an initial sparse segmentation is obtained by RPCA through a large penalization on the ℓ_1 norm. The result is sparse, and only partial foreground pixels are correctly classified. In the second step, we incorporate the sparse classified result as the priori of the spectral clustering method. The priori is specified as some linear constraints. Suppose t points have been classified as foreground in the first step using RPCA. We set down $t-1$ independent constraints, where each constraint is a $m \times 1$ vector with m being the product of image dimensions. Let V be a $(t-1) \times m$ matrix. The kth column V_k of V has only two nonzero entries: $V_k(i) = 1$, and $V_k(j) = -1$ if pixel i and pixel j are both classified to foreground in the first step. The spectral clustering problem can be reformulated incorporating with the partial grouping constraints:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x^T L x \\ \text{subject to} \quad & \|x\| = 1, \\ & V^T x = 0, \\ & \mathbf{1}^T x = 0. \end{aligned}$$

An equivalent form is

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & x^T(\alpha I - L)x \\ \text{subject to} \quad & \|x\| = 1, \\ & U^T x = 0. \end{aligned} \tag{3.2}$$

where α is a sufficiently large constant so that $\alpha I - L$ is positive definite, and the two constraints: $V^T x = 0$ and $\mathbf{1}^T x = 0$ are combined into a more compact form $U^T x = 0$. Also, since the homogeneous constraint $U^T x = 0$ is invariant to scaling, it is plausible to approximate (3.2) by the following problem:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \frac{x^T(\alpha I - L)x}{x^T x} \\ \text{subject to} \quad & U^T x = 0. \end{aligned}$$

This maximization problem is called the affinely constrained generalized Rayleigh quotients, which can be solved efficiently according to [5]. Algorithm 11 shows the steps of this method.

Algorithm 10: stepwise RPCA and Spectral clustering for image segmentation

Input: a sequence of images $M_1, ..., M_n$, parameter $\mu, \lambda, \sigma_i, \sigma_{xy}$, number of cluster$k$
Output: a 3D binary Image $\bar{M}$

Form a data matrix M, where each column of M corresponds to a cell image;
initialize $m \times n$ **matrices** S, L, Y;

while not converge do:
  compute $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$;
  compute $S = \mathcal{S}_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y)$;
  compute $Y = Y + \mu(M - L - S)$;

end while

for each column S_j **of** S**, reshape** S_j **to be a matrix with the same dimension as the images, and stack** $S_1, ..., S_n$ **together to form a three-dimensional matrix** $\bar{S}$;
Stack M_1**,...,**M_k **together to form a three-dimensional matrix M such that the sth page of M corresponds to** M_s;
Construct the Laplacian matrix L **of** M **using the RBF kernel with bandwidth** σ_i **and** σ_{xy} **respectively;**
Solve for the k+1 eigenvectors that corresponding to the k+1 smallest eigenvalues of L;
Disregard the eigenvector corresponding to the smallest eigenvalue, and form a $(m \times n) \times k$ **matrix U using the remaining eigenvectors;**
Run k-means to cluster the rows of U into k clusters;
Use $\bar{S}$ **obtained from step 6 to classify the k clusters into foreground and background, and record the result into matrix** $\bar{M}$;

3.3.2 RPCA with graph-cut penalization

The previous two methods use both spectral clustering and RPCA to solve the problem, but it does not give great improvement to the quality of the segmentation since the two methods just put together as two separate steps. To further improve the segmentation quality, we combine spectral clustering and RPCA in a more sophisticated way. As mentioned in Chapter 2, both the spectral clustering algorithm and the RPCA algorithm can be

Algorithm 11: Spectral clustering with partial grouping constraints

Input: a sequence of images $M_1, ..., M_n$, parameter $\mu, \lambda, \sigma_i, \sigma_{xy}, \alpha$
Output: a sequence of binary images

Form a data matrix M, where each column of M corresponds to a cell image;
initialize $m \times n$ **matrices** S, L, Y**;**

while not converge do:
  compute $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$**;**
  compute $S = \mathcal{S}_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y)$**;**
  compute $Y = Y + \mu(M - L - S)$
end while
for each column S_j **of S, do:**
Construct the Laplacian matrix L_j **of** M_j **using the RBF kernel with bandwidth** σ_i **and** σ_{xy} **respectively;**

compute the partial grouping constraints:
  let J **to be the index set of nonzeros of** S_j**,** $t = |J|$**;**

  for $k = 1...t - 1$**:**
  $V_k(J(k)) = 1, V_k(J(k+1)) = -1$**, where** V_k **is the kth column of** V**;**

  end for
let $U = 0_{n \times t}$**, and compute** $U(:, 1) = \mathbf{1}$**,** $U(:, 2:t) = V$**;**
compute $P = I - U(U^T U)^{-1} U^T$**;**
compute the leading eigenvector u_j **of** $P(\alpha I - L_j)P$
end for
output $U = [u_1, ..., u_m]$**.**

considered as solving minimization problems. Particularly, spectral clustering is minimizing the graph cut which imposes penalty on the goodness of an image partition. Let X_j be an indicator vector defined in the following way:

$$X_{ij} = \begin{cases} \sqrt{\frac{|\bar{A}_j|}{|A_j|}}, & v_i \in A_j, \\ -\sqrt{\frac{|A_j|}{|\bar{A}_j|}}, & \text{otherwise}, \end{cases}$$

where $A_j \subset V$ is the subset of cell pixels in image j. In Chapter 2, the weighted inner product $X_j^T L(S_j) X_j$ is shown to be $|V| RatioCut(A_j, \bar{A}_j)$, which is proportional to the sum of edge weights between the cell cluster and the background cluster. The right partition can be found if X_j minimizes $X_j^T L(S_j) X_j$. On the other hand, the sparsity structure of S_j obtained from the previous RPCA-based method indicates the cell segmentation of jth image. The indicator vector X_j and the jth column of the matrix S serve the same purpose: defining the cell segmentation, but they have different forms. The indicator vector X_j defines the segmentation with negative entry labeled background and non-negative entry labeled cells, while S_j defines the segmentation with zero entry labeled background and non-zero entry labeled cells. Our objective is to connect the segmentation defined by S_j from RPCA with its corresponding graph-cut so that the segmentation of S_j gives rise to a small graph-cut. The way of relating graph-cut with RPCA is not trivial. We will present two solutions, which correspond to Method 9 and Method 10 presented in the rest of this Chapter.

Connecting the graph-cut and RPCA through an extra non-linear constraint: One way to connect X_j and S_j is to use the Heaviside function[4]:

$$\mathcal{H}(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$$

The indicator vector X_j corresponding to the partition defined by S_j can be expressed as a summation of Heaviside functions:

$$X_j = c\mathcal{H}(|S_j| - \epsilon) - \frac{1}{c}\mathcal{H}(\epsilon - |S_j|),$$

where $c = \sqrt{\frac{m - \|S_j\|_0}{\|S_j\|_0}}$ and ϵ is a small perturbation. Based on this observation, the measurement of the graph-cut can be incorporated into the minimization problem as an extra penalty term through the connection:

[4] One continuous approximation of $\mathcal{H}(x)$ is $\frac{1}{2} + \frac{1}{2} tanh(kx)$. Since we use an approximation with larger support than the exact Heaviside function, thresholding is required according to k in order to obtain a sparse solution.

$$
\begin{aligned}
\underset{L,S,X}{\text{minimize}} \quad & \|L\|_* + \lambda_1 \|S\|_1 + \frac{\lambda_2}{2} \sum_{j=1}^{m} {X_j}^T L^{(S_j)} X_j \\
\text{subject to} \quad & L + S = M, \\
& X_j = c\mathcal{H}(|S_j| - \epsilon) - \frac{1}{c}\mathcal{H}(\epsilon - |S_j|), \\
& \mathbf{1}^T X_j = 0, \\
& X_j^T X_j = 1.
\end{aligned}
$$

The extra term $X_j^T L^{(S_j)} X_j$ controls the quality of the output S_j, so that the partition defined by S_j minimize the graph-cut. However, the optimization problem becomes much more complicated since the nonlinear constraints are involved in the problem. To simplify the problem and use existing algorithms, we introduce an auxiliary variable P_j:

$$
\begin{aligned}
\underset{L,S,X}{\text{minimize}} \quad & \|L\|_* + \lambda_1 \|S\|_1 + \frac{\lambda_2}{2} \sum_{j=1}^{n} {X_j}^T L^{(S_j)} X_j \\
\text{subject to} \quad & L + S = M, \\
& X_j = c\mathcal{H}(|S_j| - \epsilon) - \frac{1}{c}\mathcal{H}(\epsilon - |S_j|), \\
& X_j = P_j \\
& \mathbf{1}^T P_j = 0, \\
& P_j^T P_j = 1.
\end{aligned}
$$

The partial augmented Lagrangian that takes care of the first two constraints is

$$
\begin{aligned}
l(L, S, X, Y, Z) =& \|L\|_* + \lambda_1 \|S\|_1 + \frac{\lambda_2}{2} \sum_{j=1}^{n} {X_j}^T L^{(S_j)} X_j + \sum_{j=1}^{n} \langle Y_j, M_j - L_j - S_j \rangle \\
& + \frac{\mu}{2} \sum_{j=1}^{n} \|M_j - L_j - Sj\|_F^2 + \sum_{j=1}^{n} \langle Z_j, c\mathcal{H}(|S_j| - \epsilon) - \frac{1}{c}\mathcal{H}(\epsilon - |S_j|) - X_j \rangle \\
& + \frac{\mu}{2} \sum_{j=1}^{n} \|c\mathcal{H}(|S_j| - \epsilon) - \frac{1}{c}\mathcal{H}(\epsilon - |S_j|) - X_j\|_F.
\end{aligned} \tag{3.3}
$$

Similar to the alternating direction method of multipliers (ADMM), we can relax the problem by considering it as the following problem[5]:

$$\begin{aligned} \underset{L,S,X}{\text{minimize}} \quad & l(L,S,X,Y,Z) \\ \text{subject to} \quad & X_j = P_j, \\ & \mathbf{1}^T P_j = 0, \\ & P_j^T P_j = 1. \end{aligned}$$

The unknowns L and S can be solved by the minimizers of the an unconstrained optimization problem:

$$\underset{L,S}{\text{minimize}} \quad l(L,S,X,Y,Z).$$

For X and P, we can solve them by applying the method of splitting orthogonality constraints (SOC) [14]. This method is a slight modification of ADMM [1] and split Bregman iteration [10], so it is consistent with our previous steps of solving L and S. The steps are outlined below. An auxiliary variable B_j, and a parameter r are introduced similar to the method of split Bregman iteration.

1. $X_j^{t+1} = \underset{X_j}{\text{argmin}} \; l(X_j) + \frac{r}{2}\|X_j - P_j^k + B_j^k\|_2^2.$
2. $P_j = \underset{P_j^{k+1}}{\text{argmin}} \; \frac{r}{2}\|P_j - (X_j^{k+1} + B_j^k)\|_2^2 \quad \text{subject to } \mathbf{1}^T P_j = 0 \text{ and } \|P_j\|_2 = 1.$
3. $B_j^{k+1} = B_j^k + X_j^{k+1} - P_j^{k+1}.$

The first problem is an unconstrained minimization related to the previous augmented Lagrangian, which can be solved by enforcing the first necessary condition. The second problem has a closed form analytic solution. In particular, the solution to the more general problem:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \frac{r}{2}\|u - (v+w)\|_2^2 \\ \text{subject to} \quad & Au = b, \\ & \|u\|_2 = 1, \end{aligned}$$

is provided in the following theorem [14].

[5]In this optimization problem, P does not appear in the objective function. The connection of l and P is through the equality between X and P in the first constraint.

Theorem 3.3.1. *Let A be an $m \times n$ matrix with rank k, and let $A^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$ be the QR factorization of A^T, where $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ is a $n \times n$ orthogonal matrix, and R_1 is a $k \times m$ upper triangular matrix. Then, the solution to the problem:*

$$\begin{aligned} \underset{u}{\textit{minimize}} \quad & \frac{r}{2}\|u - (v + w)\|_2^2 \\ \textit{subject to} \quad & Au = b, \\ & \|u\|_2 = 1, \end{aligned}$$

is

$$u^* = \sqrt{1 - \|u_1\|_2^2}\frac{u_2}{\|u_2\|_2} + u_1,$$

where $u_1 = \underset{Au=b}{\operatorname{argmin}} \ \|u\|_2$ *and* $u_2 = Q_2 Q_2^T (v + w)$.

The proof of Theorem 3.3.1 can be found in [14]. We outline the steps of this method in Algorithm 12.

Connecting the graph-cut and RPCA trough an extra discrepancy measurement in the objective function: The previous method links the principal component pursuit (PCP) and the graph-cut problem by enforcing an equivalent relation of X_j and a nonlinear transformation of the minimizer of S_j. As mentioned in Chapter 2, since a relaxation has been applied by minimizing the augmented Lagrangian, feasibility does not fulfill exactly. However, the method imposes a price of not being feasible through some penalty terms in the augmented Lagrangian. Similarly, we can relate PCP to the graph-cut problem by adding a penalty term measuring similarity between X_j and S_j in the objective function:

$$\begin{aligned} \underset{L,S,X}{\text{minimize}} \quad & \|L\|_* + \lambda_1 \|S\|_1 + \frac{\lambda_2}{2}\sum_{j=1}^{n} {X_j}^T L^{(S_j)} X_j + \sum_{j=1}^{n}\sum_{i=1}^{m} A_{ij}^+ \\ \text{subject to} \quad & L + S = M, \\ & A = \mathbf{1} - Xsign(|S| - \epsilon), \\ & \mathbf{1}^T X_j = 0, j = 1, ..., n, \\ & X_j^T X_j = 1, j = 1, ..., n. \end{aligned} \tag{3.4}$$

Note:

$$A_{ij}^+ = \begin{cases} A_{ij}, & \text{if } A_{ij} > 0 \\ 0, & \text{otherwise} \end{cases}$$

The jth column of A connects the partition defined by X_j and the partition defined by S_j. The second constraint of the above optimization problem is to enforce that the locations of zeros in S are the same as the locations of negative entries in X. We use ADMM again to solve this problem. The augmented Lagrangian is:

$$\begin{aligned} l(L,S,X,Y,Z,A,\alpha,\beta) =& \|L\|_* + \lambda_1\|S\|_1 + \frac{\lambda_2}{2}\sum_{j=1}^{n} {X_j}^T L^{(S_j)} X_j \\ &+ \sum_{j=1}^{n}\sum_{i=1}^{m} A_{ij} + \sum_{j=1}^{n}\langle Y_j, M_j - L_j - S_j\rangle + \frac{\mu}{2}\sum_{j=1}^{n}\|M_j - L_j - Sj\|_F^2 \\ &+ \sum_{j=1}^{n}\langle Z_j, \mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\rangle + \frac{\mu}{2}\sum_{j=1}^{n}\|\mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\|_F^2 \\ &+ \sum_{j=1}^{n}\langle \alpha_j, -\mathbf{1}^T X_j\rangle + \frac{\mu}{2}\sum_{j=1}^{n}\|\mathbf{1}^T X_j\|_F^2 \\ &+ \sum_{j=1}^{n}\langle \beta_j, \mathbf{1} - X_j^T X_j\rangle + \frac{\mu}{2}\sum_{j=1}^{n}\|\mathbf{1} - X_j^T X_j\|_F^2. \end{aligned} \tag{3.5}$$

According to ADMM, we solve the problem by iteratively imposing the first necessary condition of l with respect to L, S,X,A, and then updating the Lagrange multiplier Y, Z, α, β. However, for solving a graph-cut like problem in the direction of X_j, we can avoid solving a nonlinear equation by identifying it as a normalized cut problem with extra linear constraints. A small modification needs to be applied to ADMM to suit our setting. First, we solve

$$L = \underset{L}{\operatorname{argmin}}\ l,$$

by $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$ and solve

$$S = \underset{S}{\operatorname{argmin}}\ l,$$

by $S = solve(l^{'}(S), 0)$[6] similar to Algorithm 12.

[6]S=solve(f(S),b) denotes that S solves the equation f(S)=b

Algorithm 12: RPCA with ratio-cut penalization for image segmentation

Input: a sequence of images M_1,..., M_n, and their corresponding Laplacian matrix$Lsym_1, ..., Lsym_n$, parameter $\mu, \lambda_1, \lambda_2$

Output: a sequence of binary images

Form a data matrix M, where each column of M corresponds to a cell image;

initialize L, S, Y, X, Z, P, B**;**

compute the QR factorization of $\mathbf{1}^T = \begin{bmatrix} Q_1^{m\times 1} & Q_2^{m\times(m-1)} \end{bmatrix} \begin{bmatrix} R_1^{1\times 1} \\ 0 \end{bmatrix}$**;**

choose a vector q such that $\mathbf{1}^T q = 0$**;**

compute $\mathbf{p} = q - Q_2 Q_2^T$**;**

while not converge do:

  compute $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$**;**

  solve for S **such that** $\lambda_1 tahn(kS) - Y - \mu(M - L - S) + ZH^{'}(S) + \mu(H(S) - X)H^{'}(S) = 0$**, where** $H(S) = c\mathcal{H}(|S_j| - \epsilon) - \frac{1}{c}\mathcal{H}(\epsilon - |S_j|)$**;**

  compute $Y = Y + \mu(M - L - S)$

for each column j, do:

  solve X_j **such that** $(\lambda_2 L + (r - \mu)I)X_j = Z_j + rP_j - rB_j - \mu H(S_j)$**;**

  compute $Z_j = H(S_j) - X_j$ **;**

  compute $\bar{P}_j = Q_2 Q_2^T (X_j + B_j)$**;**

  compute $P_j = \sqrt{1 - \|\mathbf{p}\|_2^2} \frac{\bar{P}_j}{\|\bar{P}_j\|_2^2} + \mathbf{p}$**;**

  compute $B_j = B_j + X_j - P_j$**;**

end while

output S;

To solve $A_{ij} = \underset{A_{ij}}{\text{argmin}}\ l$, we recognize that

$$\begin{aligned} A_{ij} =& \underset{A_{ij}}{\text{argmin}} \sum_{j=1}^{n}\sum_{i=1}^{m} A_{ij} + \sum_{j=1}^{n}\langle Z_j, \mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\rangle \\ &+ \frac{\mu}{2}\sum_{j=1}^{n} \|\mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\|_F^2. \end{aligned} \tag{3.6}$$

This problem can be solved by the following proposition [22].

Proposition 3.3.2. *The solution to the unconstrained minimization problem*

$$\underset{a}{minimize} \quad \lambda a^+ + \frac{1}{2}\|a - \gamma\|_F^2$$

is

$$\mathbf{S}_\lambda(\gamma) = \begin{cases} \gamma - \lambda, & \gamma > \lambda, \\ 0, & 0 \le \gamma \le \lambda, \\ \gamma, & \gamma < 0. \end{cases}$$

The proof of the proposition can be found in [23].

Also, note that:

$$\begin{aligned} &\langle Z_j, \mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\rangle + \frac{\mu}{2}\|\mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\|_F^2 + \frac{\|Z_j\|_2^2}{2\mu} = \\ &\frac{\mu}{2}\|\mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j + \frac{Z_j}{\mu}\|_F^2. \end{aligned} \tag{3.7}$$

Along with Proposition 3.3.2, the solution to the problem is:

$$\begin{aligned} &\underset{A_j}{\text{argmin}} \quad \frac{1}{m\mu}A_{ij}^+ + \frac{1}{2}\|\mathbf{1} + X_j sign(|S_j| - \epsilon) - A_j + \frac{Z_j}{\mu}\|_F^2 \\ &= \mathbf{S}_{(m\mu)^{-1}}(\mathbf{1} - X_j sign(|S_j| - \epsilon) + \tfrac{Z_j}{\mu}). \end{aligned} \tag{3.8}$$

To update X_j, we need to solve the following unconstrained optimization:

$$
\begin{aligned}
\underset{X_j, j=1,\ldots,n}{\operatorname{argmin}} \quad & \frac{\lambda_2}{2}\sum_{j=1}^{n} {X_j}^T L^{(S_j)} X_j + \sum_{j=1}^{n}\langle Z_j, \mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\rangle + \\
& \frac{\mu}{2}\sum_{j=1}^{n} \|\mathbf{1} - X_j sign(S_j| - \epsilon) - A_j\|_F^2 + \sum_{j=1}^{n}\langle \alpha_j, -\mathbf{1}^T X_j\rangle \\
& + \frac{\mu}{2}\sum_{j=1}^{n} \|\mathbf{1}^T X_j\|_F^2 + \sum_{j=1}^{n}\langle \beta_j, \mathbf{1} - X_j^T X_j\rangle + \frac{\mu}{2}\sum_{j=1}^{n} \|\mathbf{1} - X_j^T X_j\|_F^2.
\end{aligned} \tag{3.9}
$$

To relate (3.9) to be a graph-cut problem with linear constraints, we consider a similar problem of (3.9):

$$
\begin{aligned}
\underset{X_j, j=1\ldots n}{\text{minimize}} \quad & \frac{\lambda_2}{2} {X_j}^T L^{(S_j)} X_j + \langle Z_j, \mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\rangle \\
& + \frac{\mu}{2}\|\mathbf{1} - X_j sign(|S_j| - \epsilon) - A_j\|_F^2 \\
\text{subject to} \quad & \mathbf{1}^T X_j = 0, j = 1, ..., n, \\
& X_j^T X_j = 1, j = 1, ..., n.
\end{aligned} \tag{3.10}
$$

.

Equations (3.9) and (3.10) are not equivalent, but the constrained problem (3.10) is more accurate than the unconstrained one, and gives a better approximation to (3.4).

Let $b_j = \mathbf{1} - A_j$, and $D_j = Diag(sign(|S_j| - \epsilon))$. The above minimization problem can be rewritten in the following form:

$$
\begin{aligned}
\underset{X_j, j=1\ldots n}{\text{minimize}} \quad & \frac{\lambda_2}{2} {X_j}^T L^{(S_j)} X_j + \frac{\mu}{2} X_j^T D_j^T D_j X_j - (Z_j^T D_j + b_j D_j) X_j \\
\text{subject to} \quad & \mathbf{1}^T X_j = 0, j = 1, ..., n, \\
& X_j^T X_j = 1, j = 1, ..., n.
\end{aligned} \tag{3.11}
$$

Introducing an auxiliary variable $Q_j = -(Z_j^T D_j + b_j^T D_j) X_j$, we obtain a minimization of quadratic function with linear and spherical constraints:

$$\begin{aligned} \underset{X_j, Q_j, j=1...n}{\text{minimize}} \quad & \frac{\lambda_2}{2} {X_j}^T L^{(S_j)} X_j + Q_j \\ \text{subject to} \quad & \mathbf{1}^T X_j = 0, j = 1, ..., n, \\ & X_j^T X_j = 1, j = 1, ..., n, \\ & Q_j = -(Z_j^T D_j + b_j^T D_j) X_j. \end{aligned} \tag{3.12}$$

We can apply ADMM to the quadratic problem (3.12), and update Q_j as:

$$\underset{Q_j}{\text{argmin}} \quad Q_j + \langle \gamma_j, Q_j + (Z_j^T D_j + b_j^T Q_j) X_j \rangle + \frac{mu}{2} \| (Z_j^T D_j + b_j^T D_j) X_j + Q_j \|_F^2.$$

Since Q_j has already been taken care, we are left with the following problem:

$$\begin{aligned} \underset{X_j, j=1...n}{\text{minimize}} \quad & \frac{\lambda_2}{2} {X_j}^T L^{(S_j)} X_j + \frac{\mu}{2} X_j^T D_j^T D_j X_j \\ \text{subject to} \quad & \mathbf{1}^T X_j = 0, j = 1, ..., n, \\ & X_j^T X_j = 1, j = 1, ..., n, \\ & Q_j = -(Z_j^T D_j + b_j^T D_j) X_j. \end{aligned} \tag{3.13}$$

We can again reformulate the problem by introducing a sufficiently large constant ν so that $(\nu I - \frac{\lambda_2}{2} L^{(S_j)} - \frac{\mu}{2} D_j^T D_j)$ is positive definite. Also, the second constraint is combined in the objective function to get a more compact form. Then, (3.13) can be approximated by the following:

$$\begin{aligned} \underset{X_j, j=1...n}{\text{maximize}} \quad & \frac{X_j^T (\nu I - \frac{\lambda_2}{2} L^{(S_j)} - \frac{\mu}{2} D_j^T D_j) X_j}{X_j^T X_j} \\ \text{subject to} \quad & \mathbf{1}^T X_j = 0, j = 1, ..., n, \\ & -(Z_j^T D_j + b_j^T D_j) X_j = Q_j. \end{aligned} \tag{3.14}$$

This is the problem of affinely constrained Rayleigh quotients [5]. The solution of this type of problem is presented in [8]. We will briefly derive the solution here.

First, some notation is needed to simplify the problem. Let $B_j = \begin{pmatrix} 1^T \\ -({Z_j}^T D_j + {b_j}^T D_j) \end{pmatrix}$, and $c_j = \begin{pmatrix} 0 \\ Q_j \end{pmatrix}$. The derivation of the result requires the following lemma.

Lemma 3.3.3. *Let A be an $m \times n$ matrix. Then $Range(A) = Kernel(\bar{A}^T)$, where $\bar{A}$ is a matrix such that $Range(\bar{A}) = (Range(A))^{\perp}$.*

Proof. Let $y \in Range(A)$, then $\bar{A}^T y = 0$. Therefore, $y \in Kernel(\bar{A}^T)$. Similarly, suppose $y \in kernel(\bar{A}^T)$, $\bar{A}^T y = 0$. By the fundamental theorem of linear algebra, $y \in (Range(\bar{A}))^{\perp}$. Since $Range(\bar{A}) = (Range(A))^{\perp}$, $(Range(\bar{A}))^{\perp} = Range(A)$. Therefore, $y \in Range(A)$. □

Let K_j be a matrix such that $Range(K_j) = (Range(c_j))^{\perp}$. Since we have $B_j X_j = c_j$, $B_j X_j \in Range(c_j)$, and $K_j(B_j X_j) = 0$. Let $K_j = I_{2\times 2} - \frac{c_j c_j^T}{\|c_j\|^2}$, then $Range(K_j) = (Range(c_j))^{\perp}$ and $rank(K_j) = 1$. Let $J = \begin{pmatrix} 0 & 1 \end{pmatrix}$, then JK_j is full rank, so we have $JK_j B_j X_j = 0 \Leftrightarrow B_j X_j = c_j$. Then we can transform (3.14) into the following:

$$\begin{aligned} \underset{X_j, j=1\ldots n}{\text{maximize}} \quad & \frac{X_j^T(\nu I - \frac{\lambda_2}{2} L^{(S_j)} - \frac{\mu}{2} D_j^T D_j) X_j}{X_j^T X_j} \\ \text{subject to} \quad & JKC_j B_j X_j = 0. \end{aligned}$$

Let $W_j = JK_j B_j$, and $P_j = I - W_j^T (W_j W_j^T)^{-1} W_j$. Then the problem is equivalent to the unconstrained problem:

$$\underset{X_j}{\text{argmax}} \quad \frac{X_j^T P_j(\nu I - \frac{\lambda_2}{2} L^{(S_j)} - \frac{\mu}{2} D_j^T D_j) P_j X_j}{X_j^T X_j}, \tag{3.15}$$

which can be solved by the eigenvector corresponding to the largest eigenvalue of $P_j(\nu I - \frac{\lambda_2}{2} L^{(S_j)} - \frac{\mu}{2} D_j^T D_j) P_j$. We summarize this method in the following algorithm.

Algorithm 13: RPCA with normalized-cut penalization for image segmentation

Input: a sequence of images M_1,..., M_n, and their corresponding Laplacian matrix$Lsym_1, ..., Lsym_n$, parameter $\mu, \lambda_1, \lambda_2, \nu$

Output: a sequence of binary images

Form a data matrix M, where each column of M corresponds to a cell image;

initialize L, S, X, Y, A, Z, γ;

while not converge do:

  compute $L = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$;

  solve for S **such that** $\lambda_1 tahn(kS) - Y - \mu(M - L - S) - ZXH^{'}(S) - \mu(\mathbf{1} - XH(S) - A)XH^{'}(S) = 0$, **where** $H(S) = tanh(k(|S| - \varepsilon))$;

  compute $Y = Y + \mu(M - L - S)$

for each column j, do:

  compute $b_j = \mathbf{1} - A_j, D_j = Diag(H(S_j)), F_j = \nu I - (Lsym_j + (\frac{\mu}{2}){D_j}^T D_j)$, $B_j = \begin{pmatrix} 1^T \\ -({Z_j}^T D_j + {b_j}^T D_j) \end{pmatrix}$, $c_j = \begin{pmatrix} 0 \\ Q_j \end{pmatrix}$, $K_j = I - \frac{(c_j {c_j}^T)}{\|c_j\|^2}, J = \begin{pmatrix} 0 & 1 \end{pmatrix}$, $w_j = JK_jB_j$, **and** $P_j = I - w_j^T(w_j w_j^T)^{-1}w_j$;

  compute X_j **by computing the largest eignvector v of** $P_jF_jP_j$;

  compute $Q_j = \frac{-1}{\mu} - \frac{\gamma_j}{\mu} - ({Z_j}^T D_j + {b_j}^T D_j)X_j$;

  $\gamma_j = \gamma_j + \mu(Q_j - ({Z_j}^T D_j + {b_j}^T D_j)X_j)$;

  $A_j = S_{\frac{1}{m\mu}}(1 + \frac{Z_j}{\mu} - X_jH(S_j))$;

  $Z_j = Z_j + \mu(1 - X_jH(S_j) - A_j$;

end while

output S;

Chapter 4

Numerical Results

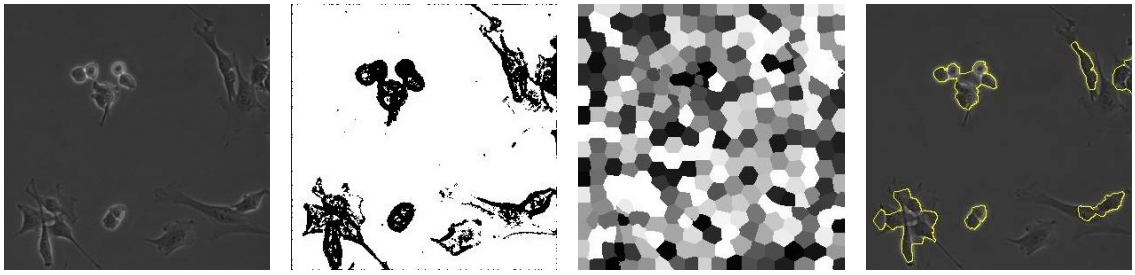
In this chapter, we illustrate the segmentation results for the methods presented in chapter 3 on a set of C2C12 cells images[1]. The code is implemented in MATLAB, so it takes a long time for the computation to complete. Due to time limitation, we down-sample the original input images to 300×300 images. The results of the hierarchical spectral clustering method and its variation will be presented in Section 4.1. The results of the RPCA based methods will be presented in Section 4.2, and the results of the RPCA based methods with graph-cut penalization will be presented in Section 4.3.

4.1 Results of the hierarchical spectral clustering method

The results of Method 1 and Method 2 applied on two cell images are shown in Figure 4.1 and Figure 4.2, respectively. The figures are displayed in a 4 columns format to describe each step of the algorithms.

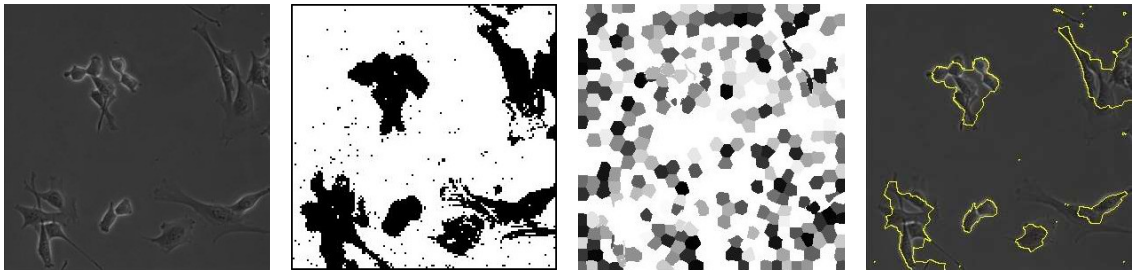
Method 1 and Method 2 are very similar. The main difference is that Method 2 performs spectral clustering on a stack of consecutive cell images, while Method 1 deals with one image at a time. Figure 4.1(a) and Figure 4.2(a) are the original 300×300 input images. The intensity values are almost the same between some cell interiors and background. They are difficult to differentiate even for human vision system. Figure 4.1(b) and Figure 4.2(b) are the two-classes spectral clustering results. We have obtained this result by performing spectral clustering to label the pixels (voxels) into two groups: cells and background. As

[1] Images of mouse myoblast cells from the Ontario Institute for Cancer Research and the Department of Medicine and Human Genetics at McGill University



(a) (b) (c) (d)

Figure 4.1: Method 1 applied on a C2C12 cell image: (a) 300×300 input image. (b) Two-classes spectral clustering result. (c) 300-classes spectral clustering result. (d) Final segmentation result.



(a) (b) (c) (d)

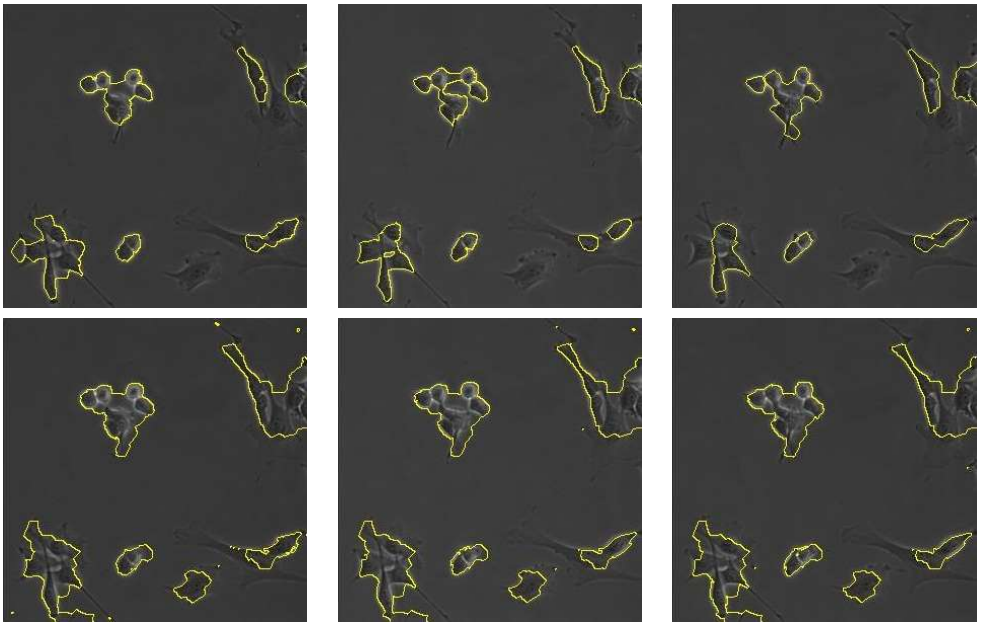
Figure 4.2: Method 2 applied on a C2C12 cell image: (a) 300×300 input image (b) Two-classes spectral clustering result. (c) 400-classes spectral clustering result. (d) Final segmentation result.

described in Chapter 3, the eigenvector associated with the second smallest eigenvalue is divided through the zero crossing. The two-classes clustering result is very sensitive to the parameters σ_i and σ_{xy} of the kernel function in the weight affinity matrix. In addition, the parameters need to be modified in every different image to obtain a good result. For image (b) in Figure 4.1, we have used $\sigma_i = 0.003, \sigma_{xy} = 100000$. Figure 4.1(c) and Figure 4.2(c) are the k-classes spectral clustering results with $k = 200$ in the 2D case and $k = 400$ in the 3D case. The number k needs to be large enough to capture fine details of the cells. Specifically, we have found out that the larger k is, the better the segmentation result. However, the problem is that the method is very time consuming for a large number of k. The computation of the 200-classes spectral clustering takes around 4909 seconds, and the 400-classes spectral clustering takes around 9698 seconds in MATLAB. Figure 4.1(d)

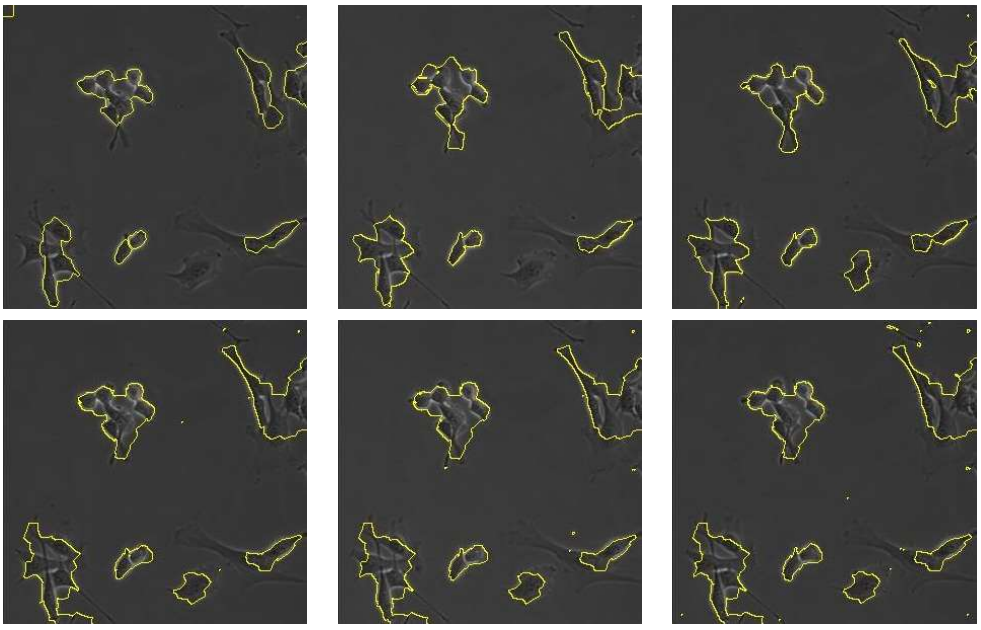
and Figure 4.2(d) show the final segmenting curves on top of the input images. To obtain the segmenting curve, we first use the two-classes spectral clustering results to determine which of the k segments of the k-classes spectral clustering results are cell segments, and background segments. Then the segmenting curves are obtained by the boundaries of the groups of the cell segments.

Method 2 gives a better segmentation result than Method 1. It uses extra information from previous image and the image after it. We compare the segmentation results of these two approaches in Figure 4.3. The first row of Figure 4.3(a) and Figure 4.3(b) show the segmentation results obtained from Method 1. The second row of Figure 4.3(a) and Figure 4.3(b) show the segmentation results obtained from Method 2. Method 1 and Method 2 both give under-segmentation of the cells. However, Method 2 often captures more cell interiors than Method 1. For example, the cell at the South-East corner are very close to the background. In our experiment, Method 1 cannot capture this cell in 5 out of 6 frames, but Method 2 can capture this cell in all 6 frames.

Method 3 is the recursive spectral clustering method for image segmentation. It performs spectral clustering recursively to identify misclassified points left from Method 1. Figure 4.4 shows Method 3 in each step. Figure 4.4(a) is the input cell image. Figure 4.4(b) shows the initial segmentation result obtained from Method 1. We mark segmented cells as one class, and perform spectral clustering again on the same image. We stop after two repetitions of spectral clustering. Figure 4.4(c) is the two-classes obtained in the second time of two-classes spectral clustering, and Figure 4.4(d) is the 200-classes obtained in the second time of 200-classes spectral clustering. Figure 4.5 compares the segmentation results of the initial segmentation results obtained using Method 1 and the results of Method 3 (the recursive spectral clustering method). Method 1 always gives under-segmentation results. Method 3 is able to capture more cell interiors. For example, Method 3 gives a great improvement of the cell cluster at the top right corner. In our experiment, Method 1 cannot capture the top part of this cell cluster. In Method 3, we solve this problem by performing spectral clustering again under the assumption that the all cell-labeled obtained from Method 1 are belong to the same class. We see that the segmentation of the cell cluster is improved in Method 3. However, in Method 3, some background pixels are forced to be cell pixels because of the compact nature of segment obtained from the spectral clustering algorithm. Recall that the affinity weight matrix also incorporates pixel locations into the similarity measurement, so pixels close to each other are easier to be classified into a same class even though there are differences between their intensities. In our experiment, the group of cells near the center of the images is classified correctly in Method 1 such that there are three individual cells in the group. However, in Method 3, these three cells merge together. They are misclassified as one large cell in 5 out of 6 images.



(a)



(b)

Figure 4.3: Comparison of segmentation results from Method 1 (first row of (a) and (b)) and Method 2 (second row of (a) and (b))

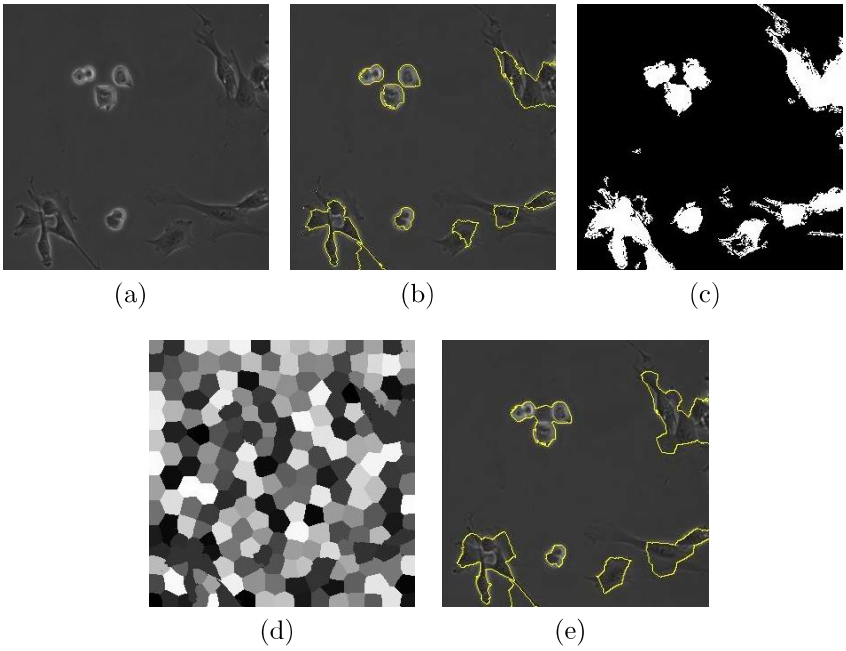
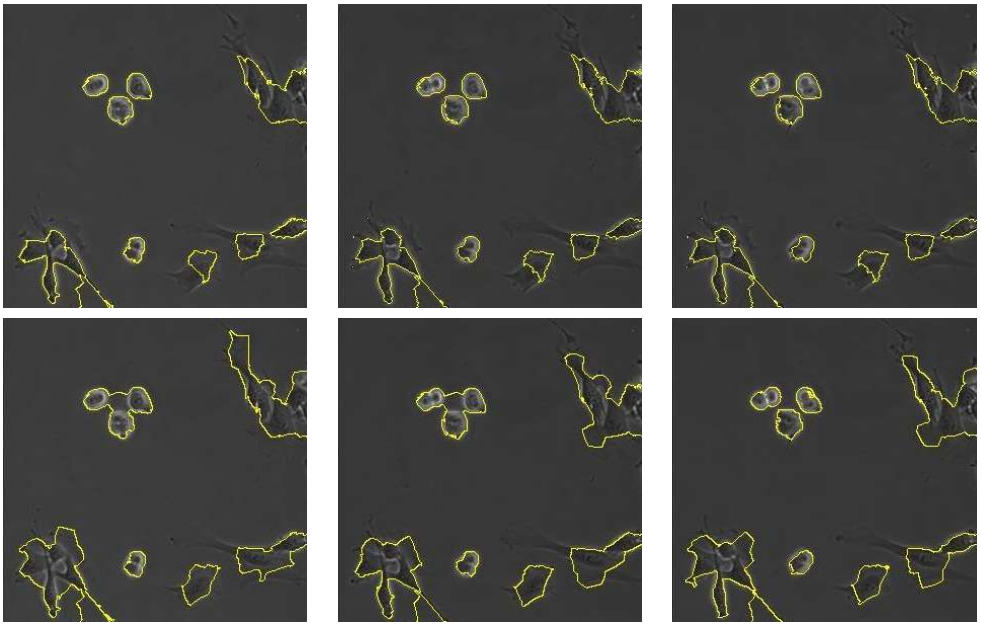


Figure 4.4: Method 3 applied on a C2C12 cell image: (a) 300×300 input image. (b) initial result obtained using Method1 (c) mask used for locating cells (d) 200 classes spectral clustering result (e)final segmentation result.

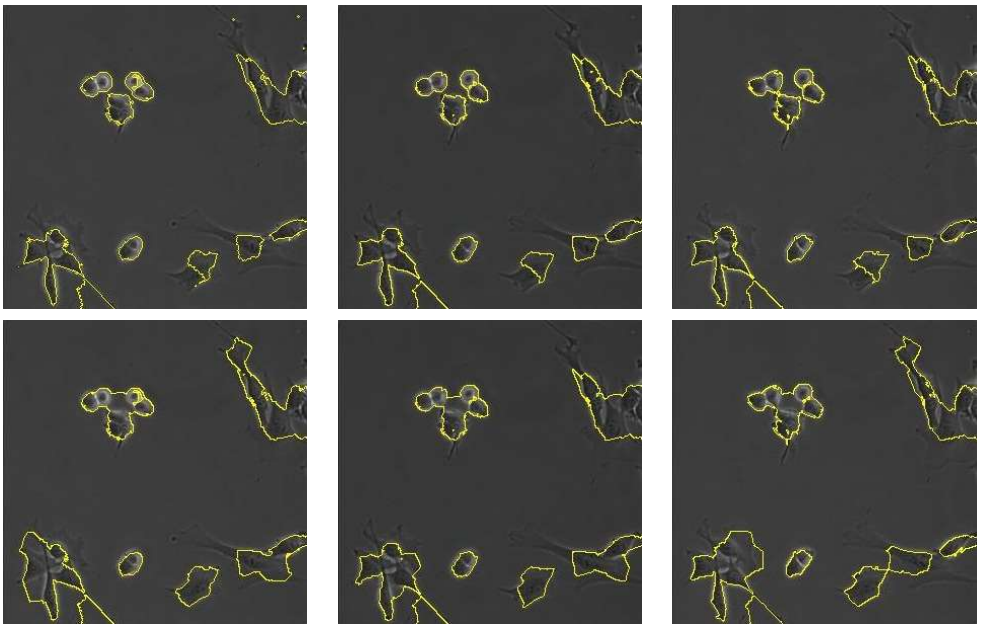
4.2 Results of the RPCA based methods

The spectral clustering methods are quite successful in capturing cell boundaries, but they are very slow due to the computation of eigenvectors and performing the k-means algorithm. The RPCA based methods are much faster than the spectral clustering based methods, and they are more robust models. We will start this section with the numerical results of the basic RPCA method introduced in Method 4 of chapter 3.

In the first stage of the algorithm, we apply RPCA on a 90000×120 data matrix M, whose columns are the cell images recording the activities of cells. We decompose the data matrix M into a sparse matrix S and a low rank matrix L using $\lambda = \frac{2}{\sqrt{m}}$ and $\mu = \frac{1}{4 \times mean(M_{ij})}$. The sparsity of S and the rank of L can be altered by changing λ and μ. The component S is more sparse for a larger λ. However, the low rank component L has larger rank for a large λ. We want S to be very sparse such that only the cell interiors are nonzero. Therefore, we set λ to be larger than the value suggested in [3]. The computation takes around half an hour, and terminates in 589 iterations. The second column of Table 4.1 shows the sparse



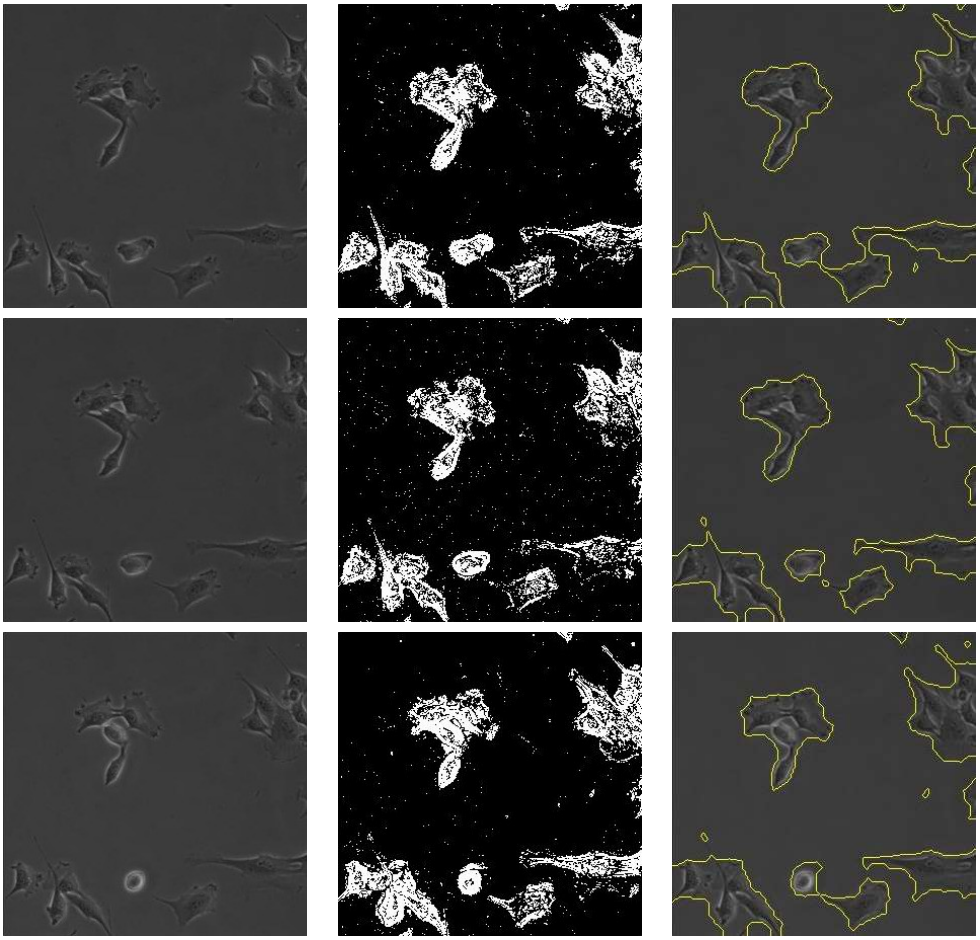
(a)



(b)

Figure 4.5: Comparison of the initial results from method 1 (first row of (a) and (b)) and the results of Method 3 (second row of (a) and (b))

component obtained from RPCA. In the second stage, we consider each column of the sparse component S as a coarse binary image by marking all nonzero entries to 1. For any image j that corresponds to the jth column of S, S_j does not give connected cell interiors. Also, there are many noises in the background. We need some post-processing as decribed in Algorithm 7 to improve the appearance of S_j. Finally, we apply the Chan-Vese active contour technique [5] on the binary image S_j to obtain the cell boundaries. The third column of Table 4.1 shows the segmentation results.



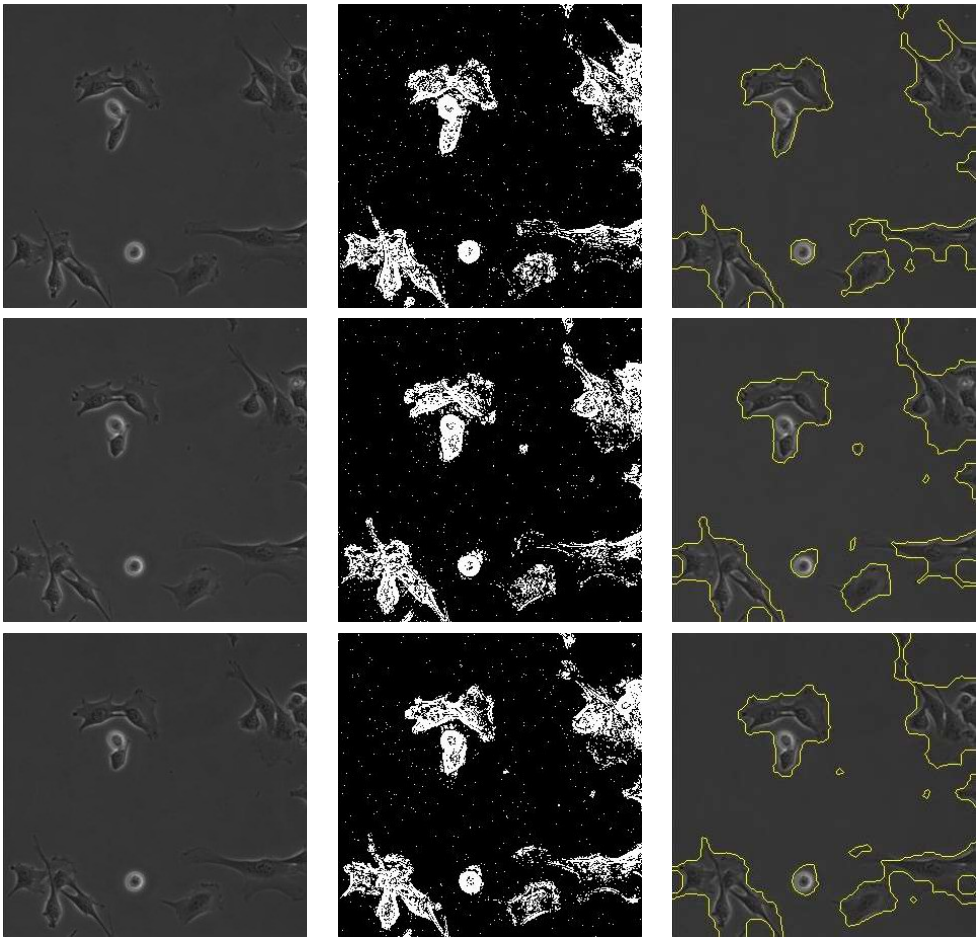
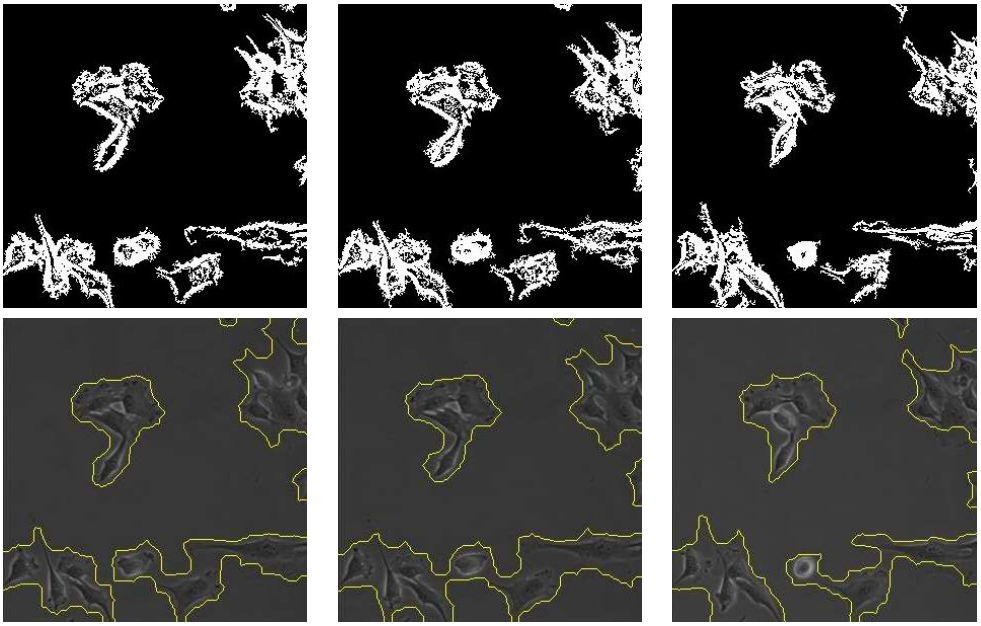
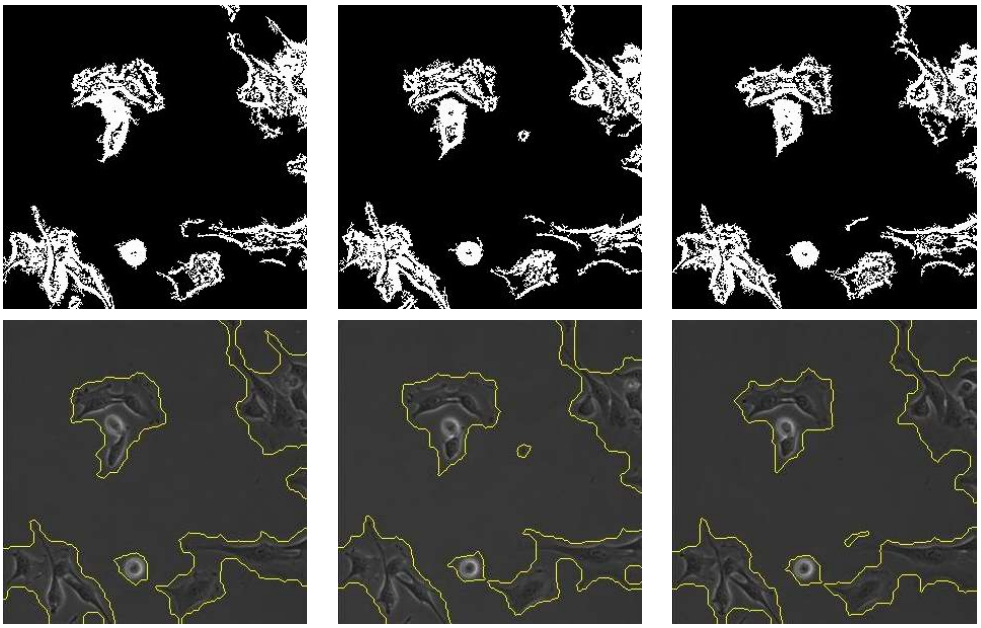


Table 4.1: Results of the RPCA method. First column: the input images. Second column:the results from PCP. Third column: the segmentation results

Some background noises are scattered in the binary images obtained from RPCA. Method 5 and Method 6 aim to eliminate the noises by modifying the optimization problem. Method 5 is the modified PCP with non-negative constraints. It is capable of erasing some of the noises since most of the background noises are points with negative values. These points have very small magnitude, and they occur due to the non-uniform intensities of the background. We have solved the PCP problem with non-negative constraints on the same data matrix M. The complexity of this method is roughly the same as Method 4. Figure 4.6 shows the segmentation results; the first row of Figure 4.6 (a) and (b) show the corresponding columns of the sparse matrix S obtained from the PCP problem with non-negative constraints; the second row of Figure 4.6 (a) and (b) show the final results.



(a)



(b)

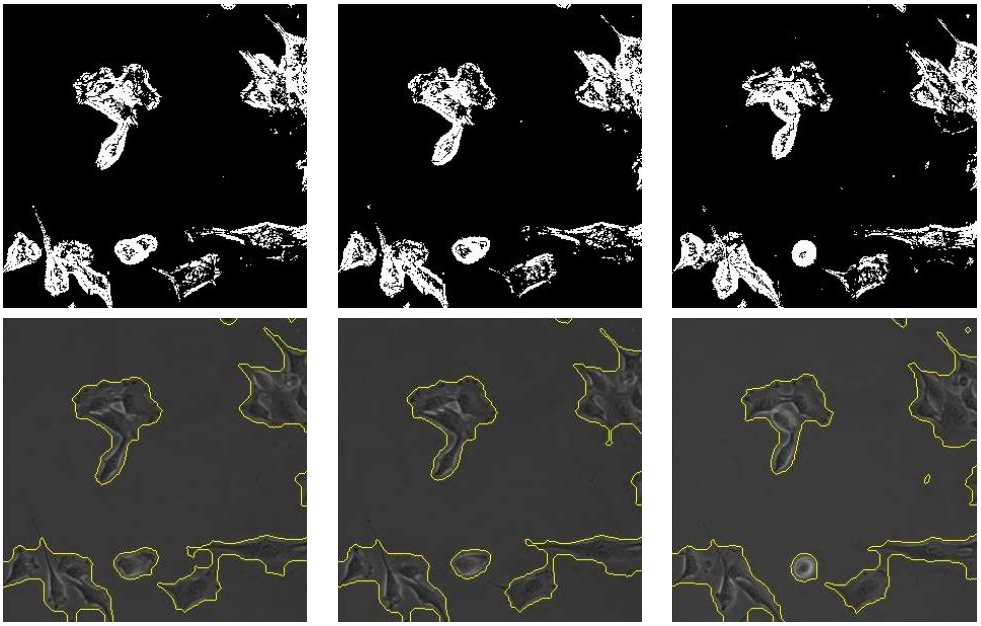
Figure 4.6: Results of the RPCA method with non-negtive constraints. First column: results from PCP with non-negative constraints. Second column: final segmentation results

Method 6 modifies the regular PCP problem to have a two-stages rescaled ℓ_1 penalty. It can therefore remove entries with small magnitude, and keep the entries with relatively large magnitude only. Figure 4.7 displays the results of the PCP with the modified ℓ_1 penalty. The method seems to be better than the other two methods in term of segmentation results, because it eliminates the background noises of the binary images from the 1st stage, while keeping the cell interiors almost as dense as the results of the regular PCP. The resulting binary images from S are almost free of background noises. We fill the holes in the cells by performing some morphological operations to get the segmentation result shown in second row of Figure 4.7 (a) and (b).

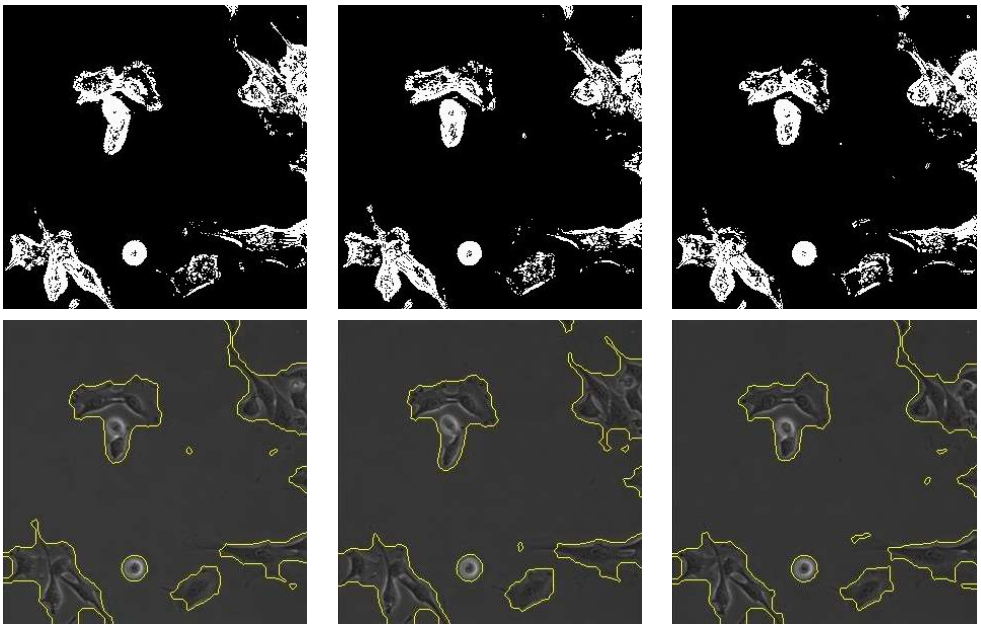
4.3 Results of the RPCA based methods with extra graph-cut penalization

One common problem of the previous three RPCA based methods is that the binary mask in the first stage cannot capture the cell interiors completely. In particular, some cell interiors break into several parts. The missing parts are almost identical to the background, so it is very hard to determine whether they are background or cells. On the other hand, the results of spectral clustering are more compact due to the inclusion of the physical distance in the affinity weight matrix. Method 9 and method 10 improve method 4 by considering extra information to evaluate the goodness of the partition. The two methods can be formulated as PCP problems with the graph-cut penalization. The only difference between Method 9 and Method 10 is the way of connecting the graph cut and the sparse component S.

Method 9 solves the PCP problem with the ratio-cut penalization. It associates the sparse component S with the ratio-cut through the Heaviside function. However, since the Heaviside function is not differentiable, we use $\frac{1}{2} + \frac{1}{2}tanh(kx)$ to approximate the Heaviside function. The parameter k controls the support of the approximation. In fact, the larger k is, the better the approximation. However, a very large k makes the zeros-finding problem in the 7th step of Algorithm 12 converge very slowly. One way to get around this problem is to apply the shrinkage $\mathcal{S}_{\frac{1}{k}}(S)$. The solution S of nonlinear equation defined in Method 9 is sparse after the shrinkage. Figure 4.8 shows the results of Method 9 applied on the same set of images. Here, we solve the modified PCP with $\mu = \frac{1}{4mean(M_{ij})}$, and $\lambda_1 = \lambda_2 = \frac{1}{\sqrt{m}}$. In this case, the computation of S is costly as it requires solving a nonlinear system. We terminate the algorithm after 167 iterations when the stopping criteria meet.

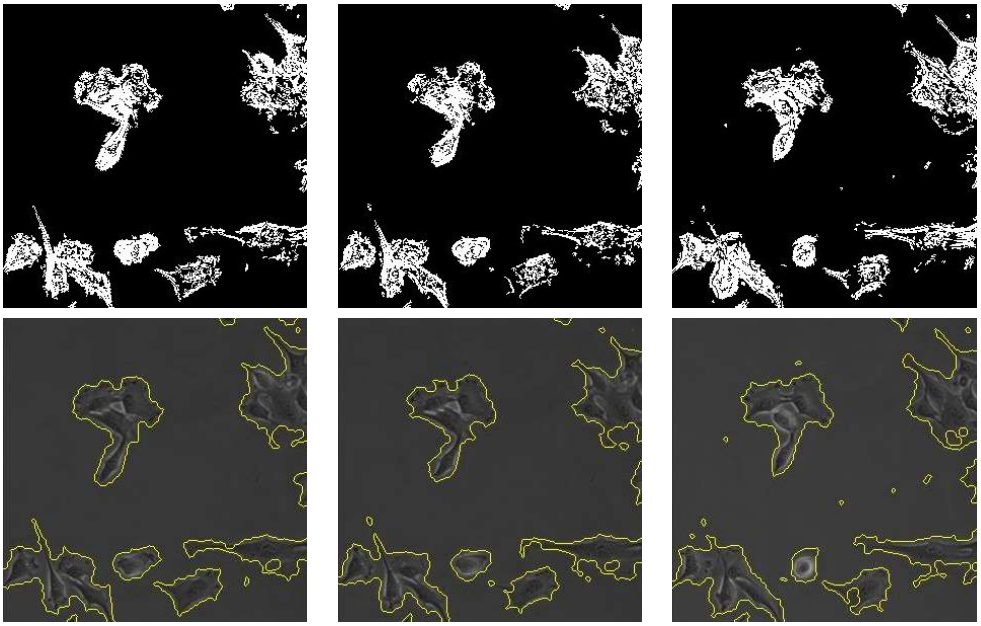


(a)

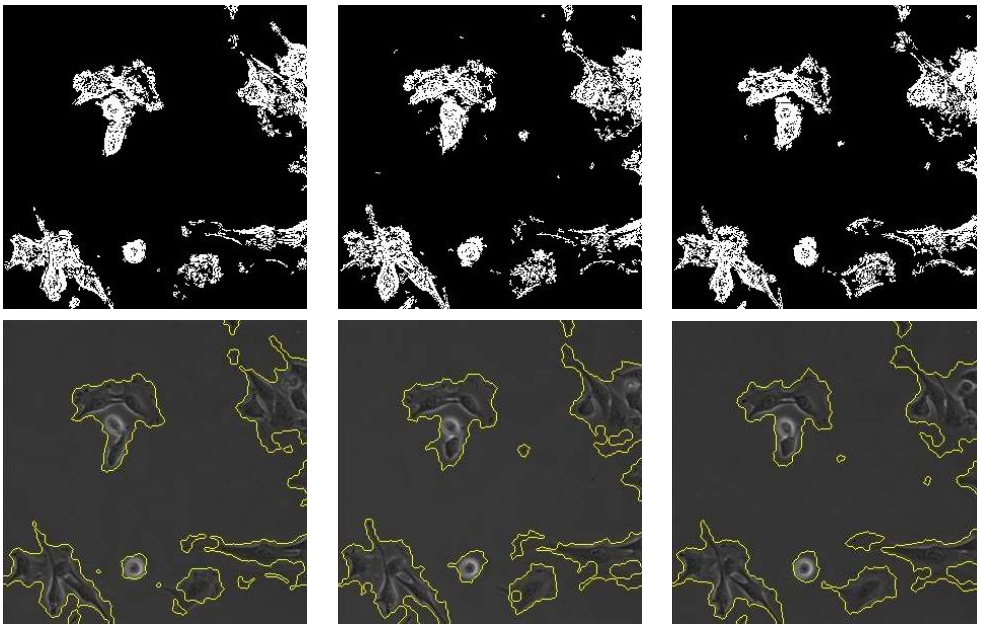


(b)

Figure 4.7: Results of the RPCA method with two-stage rescaled ℓ_1 norm. First row of (a) and (b): results from PCP with with two-stage rescaled l1 norm. Second row of (a) and (b): final segmentation results

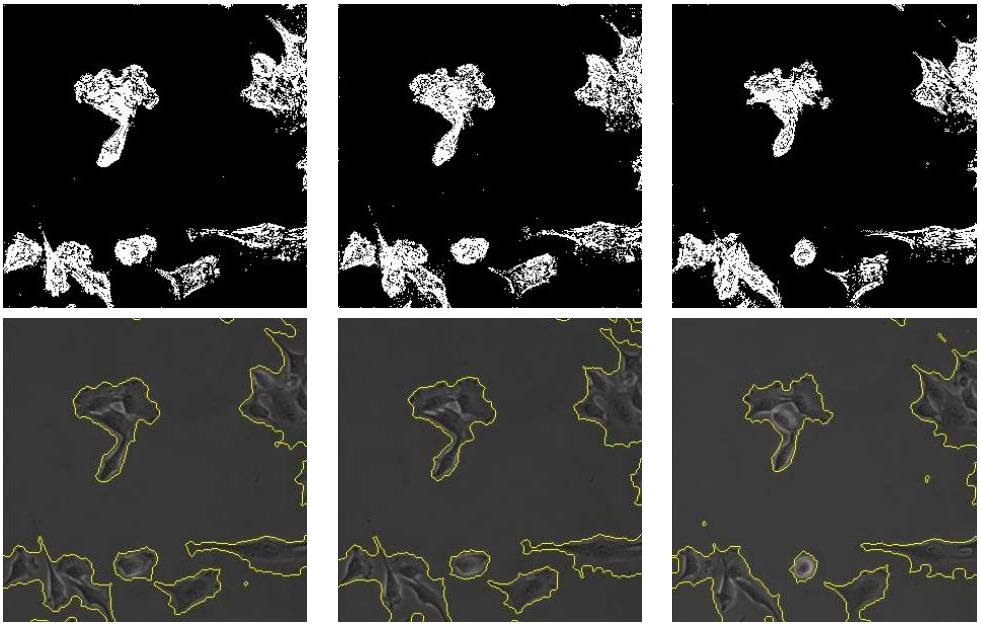


(a)

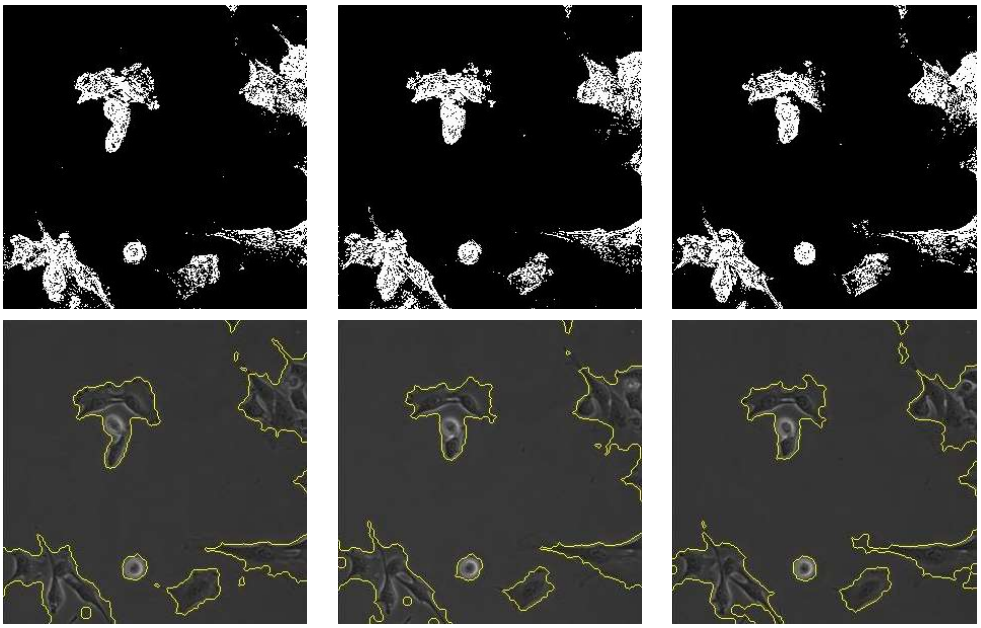


(b)

Figure 4.8: Results of the modifed RPCA method with ratio-cut penalization. First row of (a) and (b): results from PCP with with ratio cut penalization. Second row of (a) and (b): final segmentation results



(a)



(b)

Figure 4.9: Results of the modifed RPCA method with normalized-cut penalization. First row of (a) and (b): results from PCP with with normalized-cut penalization. Second row of (a) and (b): final segmentation results

Method 10 uses another way to connect the graph-cut and the sparse component S. For each image j, the model penalizes the discrepancy between the indicator vector of the normalized cut and the jth column of S. This model involves more auxiliary variables, and more parameters. However, it is not very sensitive to parameters. The method works well for reasonable range of each parameter. We solve the modified PCP problem specified in algorithm 12 with $\mu = \frac{1}{4mean(\sum_{i,j} M_{ij})}$, $\lambda_1 = \frac{1}{\sqrt{m}}$,$\lambda_2 = \frac{2}{\sqrt{m}}$, and $\nu = 10$. Figure 4.9 shows the segmentation results of this method.

More interestingly, the byproducts of Method 9 and Method 10 are the indicator variables X_j's for the jth image. They bridge the gap between PCP and the graph-cut. Theoretically, the zero crossing of X_j should define the same partition as S_j does, so it can be used to obtained the segmentation of the jth image. We have obtained very different results from Method 9 and Method 10. In Method 9, the partition defined by the indicator variables X_j is very different from the partition defined by S_j. Figure 4.10 (a) plots the partition defined by X_1^9, the first column of the matrix X obtained from Method 9. The black pixels cover all of the cells, but segmentation is coarse and cannot correctly capture the cell shapes. On the other hand, X_j from Method 10 divides the cells from the background in image j correctly. Figure 4.10 (b) plots the partition defined by X_1^{10}, the first column of the matrix X obtained from Method 10. The cell segments are compact with smooth boundaries. It gives an even better result than using S_j alone. Actually, the partition defined by X_j in Method 10 outperforms the results of all previous methods. The result is visually impressive. We present the segmentation results obtained from the indicator matrix X from Method 10 in Figure 4.11. The first row of Figure 4.11 (a) and (b) shows the partition defined by the corresponding columns of X; the second row of Figure 4.11 (a) and (b) shows the final segmentation result obtained from the partition above it.

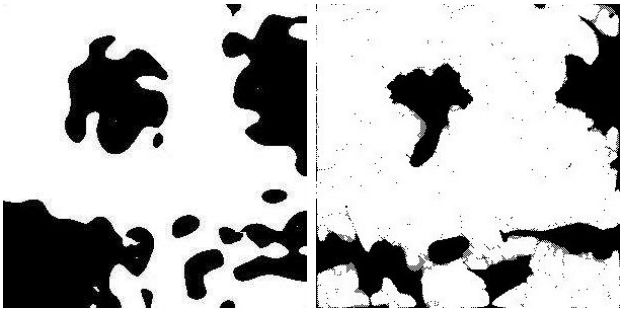
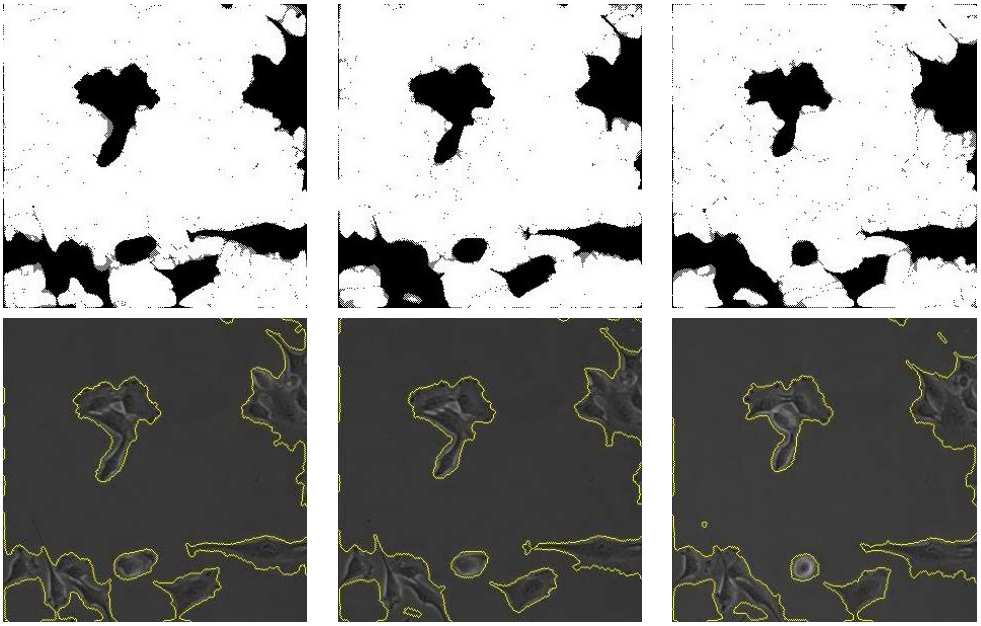
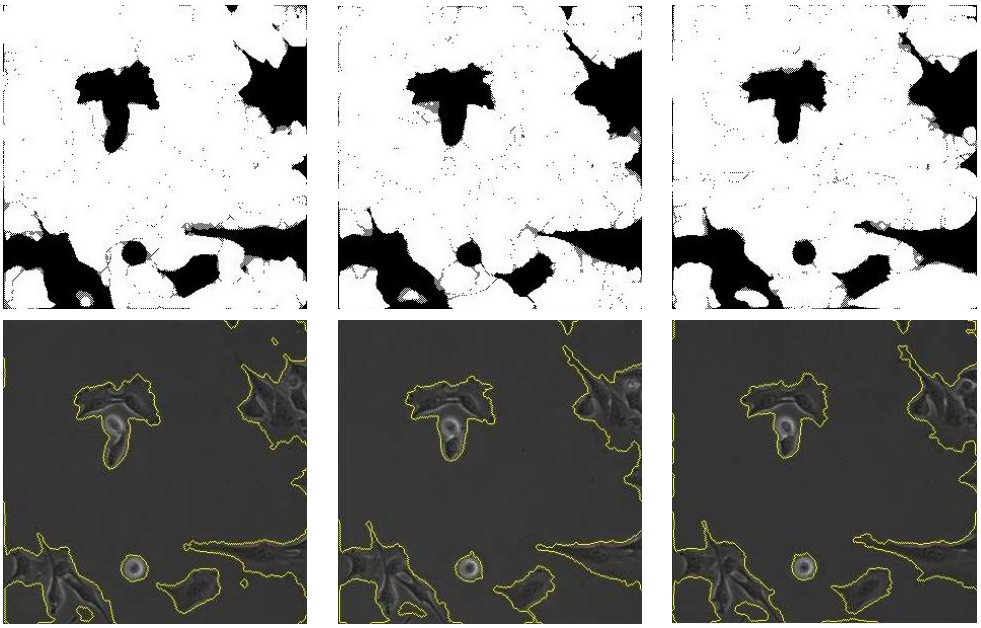


Figure 4.10: Binary segmentation results defined by X_1 from Method 9 (left) and Method 10 (right)



(a)



(b)

Figure 4.11: Results from X of the modifed RPCA method with normalized-cut penalization. First row of (a) and (b): partitions defined by corresponding columns of X. Second row of (a) and (b): final segmentation results

Chapter 5

Conclusion

This paper proposes ten methods for brightfield microscopic image segmentation. These methods are effective in capturing all of the cells in the low-contrast image, which traditional image segmentation methods cannot capture. In existing methods of brightfield microscopic image segmentation, localized steps are often required prior to global segmentation in order to obtain accurate segmentation results. In all of our methods, localized steps are no longer needed for good segmentation results.

Our spectral clustering segmentation methods give a clear and compact segmentation result, but they usually give under-segmentation of the cells. These methods are also very expensive in complexity because of the involvement in the computation of eigenvectors and k-means clustering. The RPCA-based methods, on the other hand, allow more closed-to-background cell pixels to be identified. These methods hinge on solving the principal component pursuit (PCP) problem to obtain the sparse component from the data matrix defined by the sequence of microscopic images. The computation of singular values also slow down the RPCA-based methods, but they are more efficient comparing to the spectral clustering segmentation methods. Several modifications have been applied on the PCP models to better suit the segmentation problem and improve the segmentation results. They include the modifications of extra non-negative constraints, rescale ℓ_1 norm approximation of ℓ_0 norm, and extra graph-cut penalizations. It turns out that the result produced by the method formulated as a PCP with normalized-cut penalization outperforms all other methods we have proposed. This method is more robust than the spectral clustering method since the parameters σ_i and σ_x are not as crucial in the segmentation results. Reasonable range of σ_i and σ_x usually give indifferent results. It gives clear and compact results as the spectral clustering methods do, but is also capable of capturing

more closed-to-background cell pixels as the other RPCA-based methods do.

One possible area for future research might be the incorporation of more sophisticated local grouping cues such as texture and the probability density function of each pixel into similarity measurement of spectral clustering. Also, further testing and parameters tuning is required for the spectral clustering method with partial grouping constraints. We have not provided experimental results for this method due to time limitation. The methods of associating spectral clustering to robust principal component analysis could be improved and further developed to attain better accuracy. Furthermore, the computational time could be improved by implementing the methods in parallel computing framework.

APPENDICES

Appendix A

Derivations for (2.31) and (2.32)

A.1

Show $\underset{S}{\operatorname{argmin}}\ \mathbb{L}(L,S,Y) = \mathcal{S}_{\lambda\mu^{-1}}(M-L+\mu^{-1}Y)$:

$$\underset{S}{\operatorname{argmin}}\ \mathbb{L}(L,S,Y) \quad \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}}\ \lambda\|S\|_1 + \langle Y, M-L-S\rangle + \frac{\mu}{2}\|M-L-S\|_F^2 \quad \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}}\ \lambda\|S\|_1 + \langle Y, M-L-S\rangle + \frac{\mu}{2}\|M-L-S\|_F^2 + \frac{\mu}{2}\|\mu^{-1}Y\|_F^2 \quad \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}}\ \lambda\|S\|_1 + \frac{\mu}{2}(2\langle \mu^{-1}Y, M-L-S\rangle + \|M-L-S\|_F^2 + \|\mu^{-1}Y\|_F^2) \quad \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}}\ \lambda\|S\|_1 + \frac{\mu}{2}\|M-L-S-\mu^{-1}Y\|_F^2 \quad \Leftrightarrow$$

$$\underset{S_{ij}}{\operatorname{argmin}}\ \frac{\lambda}{\mu}|S_{ij}| + \frac{1}{2}(M_{ij}-L_{ij}-S_{ij}-\mu^{-1}Y_{ij})^2 \text{ for all } i=1,...,m \text{ and } j=1,...,n. \quad \text{(A.1)}$$

Lemma A.1.1. *The solution of* $\underset{x}{\operatorname{argmin}}\ \frac{1}{2}(x-c)^2 + \nu|x|$ *is* $\mathcal{S}_\nu(c)$*, where* c *and* ν *are constants.*

Proof. Let $f(x) = \frac{1}{2}(x-c)^2 + \nu|x|$. The first necessary condition of the optimization problem is $f'(x) = x - c + \nu sign(x) = 0$. Since the objective function is convex, the

minimizer occurs at the zeros of $f'(x)$.
Case 1 $x > 0$: $x^* = c - \nu$, and $f(x^*) = c\nu - \frac{c^2}{2}$.
Case 2 $x < 0$: $x^{**} = c + \nu$, and $f(x^{**}) = c\nu - \frac{c^2}{2}$.
Case 3 $x = 0$: $x^{***} = 0$, and $f(x^{***}) = \frac{c^2}{2}$.
Therefore, we have $\underset{x}{\operatorname{argmin}}\ f(x) = \begin{cases} sign(c)(|c| - \nu), & \text{if } c > \nu \\ 0, & \text{otherwise} \end{cases} = \mathcal{S}_\nu(c)$ □

Combine result (A.1) and Lemma A.1.1, we can show that

$$S_{ij} = \mathcal{S}_{\lambda\mu^{-1}}(M_{ij} - L_{ij} + \mu^{-1}Y_{ij}),$$

which is equivalent to (2.31).

A.2

Show $\underset{L}{\operatorname{argmin}}\ \mathbb{L}(L, S, Y) = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$:

$$\begin{aligned}
&\underset{L}{\operatorname{argmin}}\ \mathbb{L}(L, S, Y) \quad \Leftrightarrow \\
&\underset{L}{\operatorname{argmin}}\ \|L\|_* + \langle Y, M - L - S\rangle + \frac{\mu}{2}\|M - L - S\|_F^2 \quad \Leftrightarrow \\
&\underset{L}{\operatorname{argmin}}\ \|L\|_* + \langle Y, M - L - S\rangle + \frac{\mu}{2}\|M - L - S\|_F^2 + \frac{\mu}{2}\|\mu^{-1}Y\|_F^2 \quad \Leftrightarrow \\
&\underset{L}{\operatorname{argmin}}\ \|L\|_* + \frac{\mu}{2}(2\langle\mu^{-1}Y, M - L - S\rangle + \|M - L - S\|_F^2 + \|\mu^{-1}Y\|_F^2) \quad \Leftrightarrow \\
&\underset{L}{\operatorname{argmin}}\ \mu^{-1}\|L\|_* + \frac{1}{2}\|M - L - S - \mu^{-1}Y\|_F^2 \qquad (A.2)
\end{aligned}$$

Lemma A.2.1. *The solution of* $\underset{X}{\operatorname{argmin}}\ \nu\|X\|_* + \frac{1}{2}\|X - W\|_F^2$ *is* $U_1\mathcal{S}_\nu[S_1]V_1^T$, *where* $W = U_1S_1{V_1}^T$ *is the singular value decomposition of* W.

Proof.

$$\underset{X}{\operatorname{argmin}}\ \frac{1}{2}\|X - W\|_F^2 + \nu\|X\|_* \quad \Leftrightarrow$$

$$\underset{X}{\operatorname{argmin}} \frac{1}{2} tr((X-W)^T(X-W)) + \nu\|X\|_* \quad \Leftrightarrow$$

$$\underset{X}{\operatorname{argmin}} \frac{1}{2}(tr(X^TX) - 2tr(X^TW) + tr(W^TW)) + \nu\|X\|_* \tag{A.3}$$

Let $X = USV^T$, and $W = U_1S_1V_1^T$ be the corresponding singular value decompositions. (A.3) is equivalent to:

$$\underset{S}{\operatorname{argmin}} (\frac{1}{2} tr(S^TS) + \nu tr(Z) + \underset{U,V}{\operatorname{argmax}} tr(W^TUSV^T)) \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}} (\frac{1}{2} tr(S^TS) + \nu tr(Z) + \underset{U,V}{\operatorname{argmax}} tr(V_1S_1^TU_1^TUSV^T)) \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}} (\frac{1}{2} tr(S^TS) + \nu tr(Z) + \underset{U,V}{\operatorname{argmax}} tr(V^TV_1S_1^TU_1^TUS)) \tag{A.4}$$

By [23], the maximum attains when $U_1 = U, V = V_1$.

$$\underset{S}{\operatorname{argmin}} (\frac{1}{2} tr(S^TS) + \nu tr(S) + tr(S_1^TS)) \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}} (\frac{1}{2} tr((S-S_1)^T(S-S_1)) + \nu tr(S)) \Leftrightarrow$$

$$\underset{S}{\operatorname{argmin}} \frac{1}{2}\|S-S_1\|_F^2 + \nu\|S\|_1 \tag{A.5}$$

By Lemma A.1.1, the minimizer is $S^* = \mathcal{S}_\nu[S_1]$. Therefore, $X = U_1S^*V_1^T = U_1S_\nu[S_1]V_1^T$. □

Combine result (A.2) and Lemma A.2.1, we can conclude that $\underset{L}{\operatorname{argmin}} \mathbb{L}(L,S,Y) = \mathcal{D}_{\mu^{-1}}(M - S + \mu^{-1}Y)$.

References

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[2] Laura Dianne Bradbury. *Segmentation of bright-field cell images*. PhD thesis, Ph. D thesis, University of Waterloo, Ontario, Canada, 2009.

[3] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011.

[4] Tony F Chan and Luminita A Vese. Active contours without edges. *Image processing, IEEE transactions on*, 10(2):266–277, 2001.

[5] Timothee Cour, Praveen Srinivasan, and Jianbo Shi. Balanced graph matching. *Advances in Neural Information Processing Systems*, 19:313, 2007.

[6] Geoff Dougherty. *Digital image processing for medical applications*. Cambridge University Press, 2009.

[7] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *Computer VisionECCV 2000*, pages 751–767. Springer, 2000.

[8] Walter Gander. Least squares with a quadratic constraint. *Numerische Mathematik*, 36(3):291–307, 1980.

[9] Gilles Gasso, Alain Rakotomamonjy, and Stéphane Canu. Recovering sparse signals with a certain family of nonconvex penalties and dc programming. *Signal Processing, IEEE Transactions on*, 57(12):4686–4698, 2009.

[10] Tom Goldstein and Stanley Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[11] Charles Guyon, Thierry Bouwmans, El-hadi Zahzah, et al. Robust principal component analysis for background subtraction: Systematic evaluation and comparative analysis. *Principal Component Analysis, P. Sanguansat, Ed*, 2012.

[12] Michel Journée, Yurii Nesterov, Peter Richtárik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *The Journal of Machine Learning Research*, 11:517–553, 2010.

[13] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[14] Rongjie Lai and Stanley Osher. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing*, 58(2):431–449, 2014.

[15] H Lütkepohl. w handbook of matrices. 1996.

[16] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, Aug 2000.

[18] Dennis L Sun. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence.

[19] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[20] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[21] Linli Xu, Wenye Li, and Dale Schuurmans. Fast normalized cut with linear constraints. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2866–2873. IEEE, 2009.

[22] Gui-Bo Ye, Yifei Chen, and Xiaohui Xie. Efficient variable selection in support vector machines via the alternating direction method of multipliers. In *International Conference on Artificial Intelligence and Statistics*, pages 832–840, 2011.

[23] Gui-Bo Ye and Xiaohui Xie. Split bregman method for large scale fused lasso. *Computational Statistics & Data Analysis*, 55(4):1552–1569, 2011.

[24] Stella X Yu and Jianbo Shi. Segmentation given partial grouping constraints. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):173–183, 2004.