Cellular Image Segmentation using N-agent Cooperative Game Theory

by

Ian B. Dimock

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, 2015

Abstract

Image segmentation is an important problem in computer vision and has significant applications in the segmentation of cellular images. Many different imaging techniques exist and produce a variety of image properties which pose difficulties to image segmentation routines. Bright-field images are particularly challenging because of the non-uniform shape of the cells, the low contrast between cells and background, and imaging artifacts such as halos and broken edges. Classical segmentation techniques often produce poor results on these challenging images. Previous attempts at bright-field imaging are often limited in scope to the images that they segment. In this paper, we introduce a new algorithm for automatically segmenting cellular images. The algorithm incorporates two game theoretic models which allow each pixel to act as an independent agent with the goal of selecting their best labelling strategy. In the non-cooperative model, the pixels choose strategies greedily based only on local information. In the cooperative model, the pixels can form coalitions, which select labelling strategies that benefit the entire group. Combining these two models produces a method which allows the pixels to balance both local and global information when selecting their label. With the addition of k-means and active contour techniques for initialization and post-processing purposes, we achieve a robust segmentation routine. The algorithm is applied to several cell image datasets including bright-field images, fluorescent images and simulated images. Experiments show that the algorithm produces good segmentation results across the variety of datasets which differ in cell density, cell shape, contrast, and noise levels.

Acknowledgements

I would like to thank my graduate supervisor Prof. Justin W. L. Wan for all his help in writing this report. The quality of this work is a testament to his input and guidance. I would also like to thank Prof. Kate Larson for taking the time to read this paper.

Dedication

I would like to dedicate this my friends and loved ones. Your support and encouragement helped make this all possible.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Cellular Images

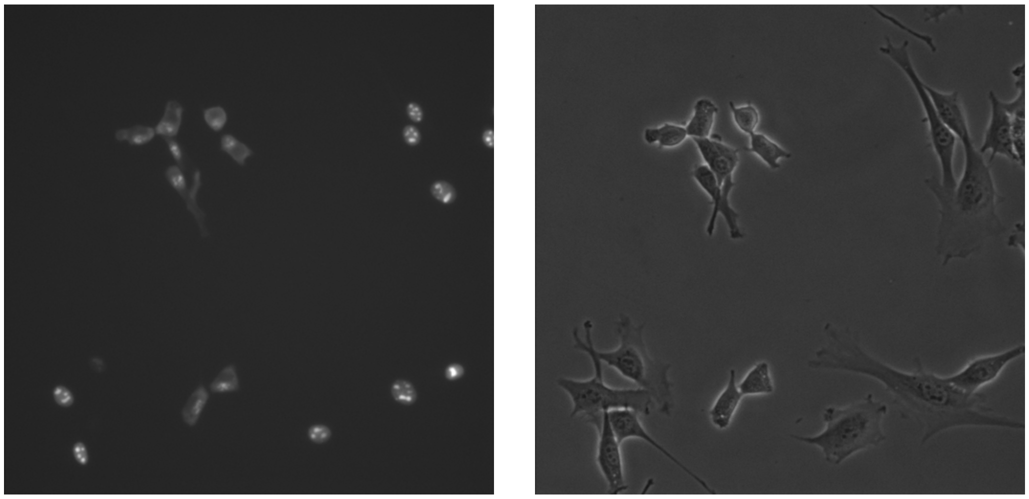
In biological science there is often a need to analyse cell images produced by microscopes. Common tasks include counting the number of cells or detecting events such as cell splitting. The analysis of these images can be done by a human or by a computer. Whether the analysis of a cellular image is to be done by hand or automatically, the type of imaging used plays an important role in the ease of the task. Two general imaging techniques are used: fluorescent and bright-field imaging. Sample fluorescent and bright-field images of the same cells can be seen in Figure 1.1.

1.1.1 Fluorescent Imaging

To take a fluorescent image, the cells must first be stained. These stains can target different parts of the cell such as the nucleus or cytoplasm. When the stained cell is exposed to a specific wavelength of light, it fluoresces at another wavelength. Only the stained parts of the cell fluoresces, and so they are easily distinguishable from the background. This high contrast is highly desirable for analysing cellular images. There are however some drawbacks to fluorescent imaging techniques. Firstly, when using many stains there can be conflicts between the wavelengths emitted by the microscope and the wavelengths fluoresced by the cells. There is also the problem of phototoxicity, which is a degradation of the stained cells with repeated exposure to the light used in the fluorescent imaging [22].

1.1.2 Bright-Field Imaging

Another type of imaging is called bright-field imaging. In this type of imaging, no stain is applied to the cells, which do not fluorescence, but rather are exposed to white light. This eliminates two of the main problems encountered in fluorescent imaging. The main disadvantage of bright-field imaging is the low contrast between cells and their background, however, inventions such as phase contrast microscopy [28] can greatly increase the contrast available in bright-field images. Bright-field images are particularly suitable for imaging living cells, and so it is desirable to have automatic segmentation algorithms which can overcome the challenging low contrast. Other artifacts present in bright-field images are glowing edges around cells called halos, as well as broken cell edges.



(a) Fluorescent Image (b) Bright-field Image

Figure 1.1: Fluorescent and bright-field images of the same cells

1.2 Image Segmentation

The problem of image segmentation is a well studied problem in computer vision. The goal is to take an image and automatically divide elements of the image into groups. These groups could be simple like foreground and background, or more complicated, such as identifying an apple and an orange in an image of a fruit basket. In this report we focus on the segmentation of cellular images. The objective for cellular images is to identify the cells from the background medium in which they are suspended.

There are two general ways to view the problem of image segmentation. If we think of an image as a grid of pixels, then the segmentation of that image is simply a labelling of the pixels as either belonging to the foreground or the background. This will implicitly define regions and boundaries in the image. This is the setting in which we develop the game theoretic segmentation algorithm in Section 3.1. Another equally valid way to view the segmentation problem is to find the boundary between foreground and background components. This method will provide implicitly defined labels for each pixel. This setting is how the Chan-Vese active contours method is developed in Section 2.4. The two viewpoints often have subtly different ways of presenting segmentation algorithms, and so keeping the correct context in mind is important when examining different methods.

1.3 Cellular Image Segmentation

There has been significant research into the area of cellular image segmentation. Most of this work has been done with fluorescent microscopy, however there has been work done with bright-field images as well. Tscherepanow et al. propose a mehtod using active contour methods for bright-field image segmentation [25]. The method performs well, but is applied to only one type of cell and so questions about its robustness remain. A different algorithm proposed by Bradbury and Wan uses a spectral k-means method [2] for bright-field image segmentation. This algorithm was also only tested with one series of images. A different approach proposed by Selinummi et al. combines multiple bright-field images of the same cells into a higher contrast projection [22]. The method is successful however requires a specific imaging technique to be used. A common problem in most cases is that the method tests only on a single dataset. The methods may not be robust for all the various issues that arise in different image datasets.

The goal for this work is to provide a cell segmentation routine that performs well on bright-field images as well as datasets from other imaging techniques. The method will ideally have minimal parameter tuning which would allow it to perform well on datasets not considered. The method uses a game theoretic approach not often used in image segmentation. Chapter 2 will introduce some mathematical background needed to develop the segmentation routine, which is presented in Chapter 3. Finally, we present some numerical results in Chapter 4 and some conclusions in Chapter 5.

Chapter 2

Background

2.1 Classical Image Segmentation Algorithms

Image segmentation is an old and important problem in computer vision, and as such, there have been many algorithms developed over the years. In the rest of this section we discuss a few of the classical image segmentation techniques. All three methods discussed behave quite differently, which gives an intuition into the many possible ways to approach the image segmentation problem. In all cases we consider the binary segmentation problem, which is the problem of segmenting an image into two classes.

The first method we will discuss is that of thresholding. In the thresholding method we simply compare the intensity of each pixel in the image with some constant threshold value. If the pixels are greater than the threshold they will receive a label of 1 and if not, then they will receive a label of 0. This is one of the simplest segmentation techniques, but is often used as a foundation for more complicated algorithms. The most obvious extension is to find a way to automatically select the threshold value [18, 21].

Another common classical technique for image segmentation is that of active contours or snakes. This technique is a form of edge detection. The goal is to find edges in an image, which then represent the boundary between the components we would like to segment. A common approach is to have this active contour be attracted to regions with large gradients in the image. This means the boundary will move to regions in the image with sharp changes in pixel intensity which represent edges. Once the active contour has settled, a segmentation can be recovered by considering the inside and outside of the curve. Beyond the gradient, many different types of information can be used to dictate the behaviour of the contour, leading to many algorithms [3, 5, 11].

The final classical technique we will discuss is that of region growing. In this method, initial seed pixels are selected in the image. From these seeds, neighbouring pixels are considered. If a neighbouring pixel is deemed a good enough fit, it is incorporated into the grouping around that seed pixel. In this way, regions grow out from the seeds until they run into each other and every pixel in the image belongs to a group. The determination of whether a pixel will join a region or not can be done in many different ways as well as the selection of the initial seeds, all of which lead to a wide variety of algorithms based on this concept [1, 10, 29].

The method we present in this paper will involve yet a different model. We will present a segmentation algorithm based on game theory. Game theory can be used in a number of different ways for segmentation [4, 8, 27], however is not a very common technique for segmentation purposes. In Section 2.2 we will introduce the necessary components of game theory before presenting the algorithm in Chapter 3.

2.2 Game Theory

The basic elements of game theory are fairly simple concepts. First is the notion of a *game*. A game is simply a set of rules within which a number of players act. These players, called *agents*, are trying to achieve the best possible reward for themselves within the rules of the game. The reward for an agent is described by an *objective function*. An agent typically has a specific set of *actions* they can perform. The objective function provides a pay-off for the agent based on his action and the actions of the other players. Game theory is the study of these games, and is concerned with predicting the outcomes while making assumptions about the behaviour of the agents.

Games can be broken down and studied in many different ways. We will introduce here a very broad classification of games that will be important in the description of our algorithm. This is the distinction between a cooperative game, and a non-cooperative game. In a non-cooperative game the agents act self-interestedly. This means that the agents care only about maximizing their objective value and are not concerned with the objective values of the other agents. To understand the outcome of such games, game theorists often consider the concept of a Nash Equilibrium [17].

A Nash Equilibrium in a game is a set of strategies for all agents in the game such that no agent would improve their objective value by taking a different strategy. In this context a strategy represents a distribution over the actions available to an agent. A Nash Equilibrium defines strategies for rational agents in that to maximize ones objective value an agent playing a game with other rational agents should play the strategy dictated by the Nash Equilibrium. John Nashed proved that in a finite game with rational agents playing strategies which are distributions over actions, a Nash Equilibrium is guaranteed to exist, although the proof is non-constructive. In practice, Nash Equilibria can be difficult to find.

The Nash Equilibrium for a game can often be unintuitive. The classic example is of this is the Prisoner's Dilemma [16]. In this game, the Nash Equilibrium represents an outcome where two prisoners betray one another and end up both serving longer sentences than if both had stayed silent. If both prisoners had kept quiet, they would both have received a lesser sentence than that identified as the Nash Equilibrium. The reason why this happens is because the agents are acting only for themselves, regardless of the other prisoners decision, confessing will improve a prisoners objective value. Had they been cooperating, the agents would have chosen to both stay silent. Allowing cooperation leads to a new paradigm in game theory, which is the study of cooperative games.

In a cooperative game, the agents are allowed some form of cooperation. The type of cooperation allowed can be quite varied. One common form of cooperative game allows agents to form coalitions. By acting together in a coalition, the agents can often improve their objective values over what they would have achieved playing self-interestedly. Determining how agents will form coalitions is another area of study within game theory named the coalition structure generation problem.

2.3 K-means

The k-means algorithm [13] is a clustering method used to group a sequence of n vectors, $(\boldsymbol{z}_1, \boldsymbol{z}_2, \dots \boldsymbol{z}_n)$ into k distinct classes $\boldsymbol{S} = (S_1, S_2, \dots S_k)$, where each vector is $\boldsymbol{z}_i$ is present in one and only one class S_j. To split the vectors, the k-means algorithm attempts to minimize the within-cluster sum of squares according to the following formula:

$$\arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{\boldsymbol{z} \in S_i} \|\boldsymbol{z} - \boldsymbol{\mu}_i\|^2, \tag{2.1}$$

where $\boldsymbol{\mu}_i$ is the within cluster mean:

$$\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\boldsymbol{z} \in S_i} \boldsymbol{z}. \tag{2.2}$$

This is a hard optimization in general, and so the k-means algorithm does not guarantee an optimal solution. The k-means algorithm is initialized by picking k starting centres. These are the vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ that will be compared with our input vectors. These can be chosen a number of ways, one of which is to randomly select k of the input vectors to become the centres.

The algorithm then proceeds by alternating two steps: the assignment step, and the update step [12]. In the assignment step, each input vector $\boldsymbol{z}_i$ is compared to each centre $\boldsymbol{\mu}_j$. Each vector is then assigned to whichever class had the centre with which it had the least Euclidean distance. After the assignment step, we proceed with the update step. In the update step, the new centres $\boldsymbol{\mu}$ are computed using equation (2.2).

The k-means algorithm iterates until no more changes are made in an assignment step. The result is a clustering of the n vectors into k classes.

2.4 Chan-Vese Active Contours

The active contours method of Chan and Vese [5, 24] is an active contours segmentation algorithm. The algorithm attempts to find a deformable boundary between foreground and background components. This is a different way to view the segmentation problem than the labelling of pixels we will use to describe the main algorithm in Chapter 3, however they are interchangeable. We introduce this model as it is used as a post-processing step in the main algorithm.

The algorithm uses a deformable curve, D, defined by the zero level set of a function ϕ, $D = \{(x, y) \mid \phi(x, y) = 0\}$. This curve represents a segmentation; all pixels (i, j) inside the curve will have $\phi_{i,j} < 0$, and all pixels outside the curve will have $\phi_{i,j} > 0$. To find ϕ, the algorithm solves the following optimization problem:

$$\underset{\mu_0, \mu_1, D}{\arg\min}\, F(\mu_0, \mu_1, D), \tag{2.3}$$

where F is the functional:

$$\begin{aligned} F(\mu_0, \mu_1, D) = &\ \rho \cdot \text{Length}(D) \\ &+ \lambda_1 \int_{inside(D)} |u(x,y) - \mu_0|^2 dx\ dy \\ &+ \lambda_2 \int_{outside(D)} |u(x,y) - \mu_1|^2 dx\ dy. \end{aligned} \tag{2.4}$$

Here ρ is a model parameter that controls the strength of the length regularization term. λ_1 and λ_2 are also model parameters but are often set to 1. μ_0 is the average pixel intensity of the image u inside of D and similarly μ_1 is the average pixel intensity outside of D.

The main components of this functional are the two integral terms. These two terms compare pixels inside or outside of the curve with the average pixel intensity in the respective region. If a pixel inside the curve has intensity very far from the average pixel intensity inside the curve, the pixel will exert a stronger influence on the boundary, moving it such that the pixel will be outside the boundary. In normal active contours methods, gradient information is used to determine when to stop changing the boundary. The boundary in the Chan-Vese algorithm stops changing when a balance is reached between the intensities of the pixels inside and outside the curve. This process is much less local than relying on gradient information and in general produces good segmentation results.

The other term in the functional controls for the length of the boundary; ρ is a scaling parameter that can be adjusted to allow for a long winding boundaries, or more compact circular ones. The value of ρ will often depend on the types of images being segmented.

From the optimization problem, one can derive the Euler-Lagrange equations and from those a numerical approximation of the PDE can be made. The resulting discretization is:

$$\begin{aligned} \frac{\phi_{i,j}^{n+1} - \phi_{i,j}^n}{\Delta t} = \delta_h(\phi_{i,j}^n) \Bigg[& \frac{\rho}{h^2} \Delta_-^x \cdot \left(\frac{\Delta_+^x \phi_{i,j}^{n+1}}{\sqrt{(\Delta_+^x \phi_{i,j}^n)^2/(h^2) + (\phi_{i,j+1}^n - \phi_{i,j-1}^n)^2/(2h)^2}} \right) \\ & + \frac{\rho}{h^2} \Delta_-^y \cdot \left(\frac{\Delta_+^y \phi_{i,j}^{n+1}}{\sqrt{(\phi_{i+1,j}^n - \phi_{i-1,j}^n)^2/(2h)^2 + (\Delta_+^y \phi_{i,j}^n)^2/(h^2)}} \right) \\ & - \lambda_1 (u_{i,j} - \mu_0(\phi^n))^2 + \lambda_2 (u_{i,j} - \mu_1(\phi^n))^2 \Bigg], \end{aligned} \tag{2.5}$$

where h is the space step, δ_h is a regularization of the Dirac delta function, and Δ_+^x, Δ_-^x, Δ_+^y, and Δ_-^y are forward and backward differences in the x and y directions respectively. An artificial time step Δt is also introduced. The function δ_h is defined as follows:

$$\delta_h(z) = \frac{1}{z^2+1}. \tag{2.6}$$

This function is used to ensure that ϕ changes only near the boundary (the zero level set of ϕ). The four finite difference operators are defined as follows:

$$\Delta_+^x \phi_{i,j} = \phi_{i+1,j} - \phi_{i,j}, \tag{2.7}$$
$$\Delta_-^x \phi_{i,j} = \phi_{i,j} - \phi_{i-1,j}, \tag{2.8}$$
$$\Delta_+^y \phi_{i,j} = \phi_{i,j+1} - \phi_{i,j}, \tag{2.9}$$
$$\Delta_-^y \phi_{i,j} = \phi_{i,j} - \phi_{i,j-1}. \tag{2.10}$$

We can solve this implicit discretization by alternatingly solving for ϕ^{n+1}, and updating μ_0 and μ_1. We stop the iterations after a maximum of T steps or if ϕ has sufficiently converged.

2.5 ISBI 2013 Cell Tracking Challenge

The IEEE International Symposium on Biomedical Imaging 2013 Cell Tracking Challenge[1] was held to establish methods for evaluating different automatic cell tracking algorithms [15]. There were eight different datasets used in the competition and six competing algorithms. As part of the evaluation, segmentation quality was measured. It is this segmentation quality we wish to use when evaluating our algorithm described in Chapter 3.

2.5.1 Datasets

The eight datasets can be divided into real videos and simulated videos, and also 2D and 3D videos. The real videos are obtained from a variety of imaging techniques. Our efforts

[1]Challenge website: http://www.codesolorzano.com/celltrackingchallenge/Cell_Tracking_Challenge/Results_First_CTC.html

focus only on the 2D images. For each dataset, a number of ground truths are provided. Due to the high number of frames and cells, however not all cells or frames have ground truths provided. At least one frame in each dataset is guaranteed to be fully segmented, while cells in other frames are chosen randomly to be given labels in the ground truth. Considering only the 2D videos, we are left with four datasets which are described below. Sample frames are shown in Figure 2.1.

C2DL-MSC

One of the real video datasets. The cells are rat mesenchymal stem cells. The video consists of 2D images taken at low resolution using cytoplasmic labelling. Two different time series are given for this cell type, having sizes 992×832 and 1200×782 respectively. Segmentation challenges include a high signal-to-noise ratio and cells often having long protrusions.

N2DH-GOWT1

One of the real video datasets. The cells are GOWT1 mouse embryonic stem cells. The video consists of 2D images taken at high resolution using nuclear labelling. Two different time series are given for this cell type, both having size 1024×1024. Segmentation challenges include variations in staining intensity and cell splitting.

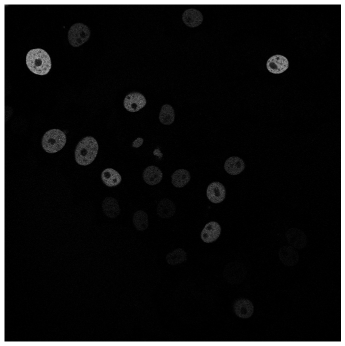
N2DL-HeLa

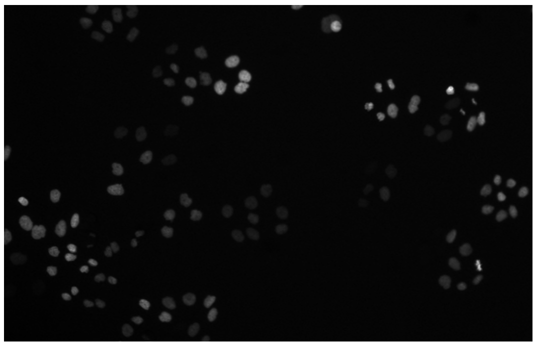
One of the real video datasets. The cells are histone 2B expressing HeLa cells. The video consists of 2D images taken at low resolution using nuclear labelling. Two different time series are given for this cell type, both having size 1100×700. Segmentation challenges include the large number of cells and frequent cell splitting in the images.

N2DH-SIM

One of the simulated video datasets. The simulated videos are generated using a simulation toolkit [23]. Six different time series are given for this cell type, the largest of which has size 510×580. The variations amongst the simulations provide different challenges to segmentation, including different noise levels and cell densities.

(a) C2DL-MSC


(b) N2DH-GOWT1


(c) N2DL-HeLa

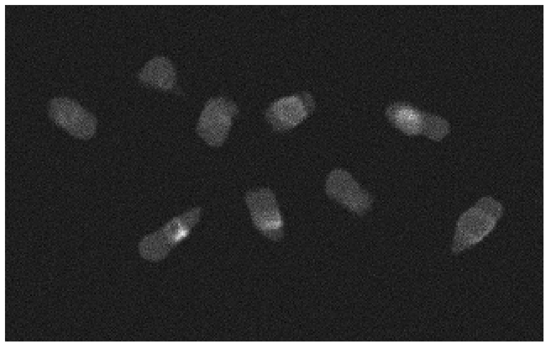

(d) N2DH-SIM

Figure 2.1: Sample frames from ISBI 2D videos

2.5.2 Submissions

Six groups submitted algorithms to the cell tracking challenge. Below we briefly describe the segmentation algorithm used by each.

COM-US

Segmentation is performed by first computing an intensity histogram. It proceeds by iteratively computing a best-fit N-point histogram. The middle value of this N-point histogram provides a threshold for segmentation [6].

HEID-GE

Before segmentation a Gaussian filter is applied. Next an initial segmentation was achieved using adaptive region thresholding. Finally the segmentation is cleaned up using median filtering and hole filling [9].

KTH-SE

A smoothed image and a background estimation are produced by convolving two different Gaussian kernels with the original image. After a band pass filter with these two new images a global thresholding is done to produce a segmentation [14].

LEID-NL

In this algorithm segmentation and tracking are performed simultaneously. At each time step each cell is fit with a model which evolves over time. The models are fit by performing gradient descent on an energy functional encapsulating different image data [7].

PRAG-CZ

The image is first smoothed by Gaussian filter. Segmentation is performed using an iterative thresholding based on k-means. Thresholding is computed on a sliding window to correct for differences in different parts of the image [19].

UPM-ES

This algorithm uses spatio-temporal filtering followed by label clustering based on histogram analysis [20].

Chapter 3

Methodology

The segmentation of cellular images presents various challenges that classical segmentation techniques often fail to overcome. Two challenges in particular are the low contrast between foreground and background pixels as well as the noise introduced by the microscopic imaging techniques. For these reasons, novel approaches are required to effectively segment cellular images.

In this chapter we outline the algorithm we use to perform cellular image segmentation. We frame the segmentation problem as a game theoretic one, with each pixel in the image acting as an agent. For cellular image segmentation we are required to divide the image into foreground and background - the cells, and the medium in which they are situated. The pixels acting as agents must choose between two strategies representing the foreground and background. When all agents have reached a consensus on their respective strategies, we are left with our final segmentation.

The game theoretic portion of the algorithm is detailed in Section 3.1. It is based on work by Guo, Yu and Ma [8]. It consists of a non-cooperative component described in Section 3.1.2, as well as a cooperative component described in Section 3.1.3. In the non-cooperative game, agents act self-interestedly and rely only on local information to make their decisions. In the cooperative game, agents are allowed to form coalitions. By acting together in a coalition, agents can improve the coalition-wide results to the detriment of certain agents. By coming to a consensus as a coalition the agents are acting upon information from a much larger portion of the image than in the non-cooperative game.

The algorithm proceeds by combining the results of the cooperative and non-cooperative components. This is possible because in both games, the agents use objective functions built on the same underlying energy minimization problem described in Section 3.1.1. The integration of the components is described in Section 3.1.4.

Additional components are discussed in subsequent sections. The initial splitting of foreground and background components using k-means is described in Section 3.2, and the initial configuration of coalitions is described in Section 3.3. Finally in section Section 3.4 we describe a smoothing procedure which uses a Chan-Vese active contours model on the binary image that is the output of the game theoretic segmentation routine.

3.1 N-agent Game Theoretic Image Segmentation

In this model we represent the $m \times n$ pixel image as a 2D grid of points. We index the points either by a conventional pair (i, j), with $i \in \{1 \ldots m\}, j \in \{1 \ldots n\}$ or as a single index $s \in S = \{1 \ldots N\}$ where $N = m \cdot n$. We can convert from (i, j) to s using the following formula:

$$s = m \cdot (j - 1) + i. \tag{3.1}$$

The image data is given as $u = (u_1, \ldots, u_N)$, where each u_s takes a value between 0 and 1, representing the intensity of a pixel. The goal of image segmentation is to identify foreground and background components. To achieve this we would like to assign each pixel in the image to either the background or the foreground represented by a label 0 or 1. We represent the segmentation as $w = (w_1, \ldots, w_N)$, here each w_s is a binary variable taking either 0 or 1. A segmentation w thus provides each pixel with a label giving us potential foreground and background components.

3.1.1 Game Theoretic Objective Function

There are 2^N possible segmentations of a given image with N pixels. We would like to find the best possible segmentation of our image. We call this optimal segmentation w^*. The segmentation w^* is that segmentation which is best explained by the pixels in our image. We can write this as a conditional probability $P(w|u)$. This probability represents the quality at which a given segmentation w is explained by the image u. Finding the segmentation w^* which maximizes this probability is an optimization problem with objective function $P(w|u)$. By applying Bayes' rule we arrive at the following optimization problem:

$$w^* = \arg\max_w P(w|u) = \arg\max_w P(u|w)P(w). \tag{3.2}$$

Under the random field model we are using, we have that $P(u|w)$ is independent across all points in the grid, so:

$$P(u|w) = \prod_s P(u_s|w_s). \tag{3.3}$$

We further assume that the distribution of $P(u_s|w_s)$ is Gaussian. We have two classes, foreground and background, represented by 0 and 1. Each class will be represented by its own Gaussian distribution. We specify these distributions by means and standard deviations: (μ_0, σ_0) for class 0, and (μ_1, σ_1) for class 1. We will let k represent one of the two possible labels for a pixel, 0 or 1, so $k \in \{0, 1\}$. We are now ready to write down the distribution for $P(u_s|w_s)$ for a pixel s taking class k:

$$P(u_s|w_s) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(u_s - \mu_k)^2}{2\sigma_k^2}\right). \tag{3.4}$$

To simplify our optimization we take the log of the argument. This preserves optimality since log is a monotone function. We get:

$$w^* = \arg\max_w \sum_s \Big[\log\left(P(u_s|w_s)\right)\Big] + \log\left(P(w)\right). \tag{3.5}$$

The 2D grid that we have presented with the Gaussian distribution is a Markov Random Field (MRF). It satisfies the conditions of the Hammersley-Clifford Theorem, which allows us to write $P(w)$ according to a Gibbs distribution:

$$P(w) = \frac{1}{Z} \exp\left(-\sum_{X\in\mathcal{X}} V^X(w)\right). \tag{3.6}$$

Here, Z is called the normalizing constant, and can be ignored after taking the logarithm. The function V^X is called the clique potential for a clique X. A clique is a set of pixels all of which neighbour each other. We consider only four neighbours for each pixel (up, down, left, and right) so cliques in our grid consist only of pairs of pixels. Here $\mathcal{X}$ represents the set of all cliques. We consider four neighbours for each pixel which produces only cliques of size two. This is because all pixels in a clique must be neighbours with one another. Since we only ever consider cliques of size two, we can redefine the potential function correspondingly:

$$V^X(w) \equiv v(w_s, w_r) = \begin{cases} -\beta & \text{if } w_s = w_r \\ \beta & \text{if } w_s \neq w_r \end{cases}, \text{where } X = \{s, r\}. \tag{3.7}$$

Here β is a model parameter, which controls the scaling of the clique potential.

We can express our optimization in terms of energy. We wish to find the lowest total energy E^{tot}. We express this by expanding $P(w)$ in our objective function and grouping all terms by index s to get:

$$w^* = \arg\min_w E^{tot} \equiv \arg\min_w \sum_s E(u, w, s), \tag{3.8}$$

where $E(u, w, s)$ represents the local energy for a given pixel s. It is the combination of terms from the Gaussian distribution and the clique potentials of all cliques to which s belongs. Recall, a pixel can be labelled 0 or 1 in our segmentation w. We define the energy when pixel s is labelled 0 to be $E^0(u, w, s)$. Similarly, when pixel s is labelled 1 the energy is $E^1(u, w, s)$. To minimize the total energy, we define the local energy of each pixel to be the minimum of these two:

$$E(u, w, s) = \min \left[E^0(u, w, s), E^1(u, w, s) \right]. \tag{3.9}$$

The energy $E^0(u, w, s)$ is computed using (μ_0, σ_0) from the Gaussian distribution for class 0, and is defined as follows:

$$E^0(u, w, s) = \log(\sqrt{2\pi}\sigma_0) + \frac{(u_s - \mu_0)^2}{2\sigma_0^2} + \sum_{r \in N(s)} v(0, w_r), \tag{3.10}$$

where $N(s)$ represents the neighbourhood of pixel s (all pixels belonging to a clique with s). We can similarly define $E^1(u, w, s)$:

$$E^1(u, w, s) = \log(\sqrt{2\pi}\sigma_1) + \frac{(u_s - \mu_1)^2}{2\sigma_1^2} + \sum_{r \in N(s)} v(1, w_r). \tag{3.11}$$

Equations (3.10) and (3.11) give us a way to compare the two possible labels to which we can assign to pixel s. They will be the foundation on which we build the cooperative and non-cooperative components described in the following sections.

3.1.2 Non-Cooperative Strategy

From an initial segmentation w_0, we allow the pixels (agents) to compete in a non-competitive game where each pixel tries to minimize its local energy, using the labels of its neighbours as information. By finding a Nash equilibrium amongst the agents we can achieve the desired result where no agent is able to improve its objective value by changing its label.

There are two difficulties in finding this equilibrium. Firstly, we are not allowing the pixels to pick strategies which are distributions over their actions, and as such the Nash Equilibrium is not guaranteed to exist. Secondly, even if it does exist it may be difficult to find due to the number of agents involved. We turn to a technique called best response dynamics, which is a method by which agents choose their strategies. It is an iterative procedure by which agents update their strategy based on local information. In this case, the pixels will update their label based on the labels of their neighbours. By this we mean that each pixel s will compute E^0 and E^1 based on the previous segmentation, and choose either the strategy 0 or 1 which produced the lowest energy. This process can converge to a Nash equilibrium or sometimes get stuck in a cycle. Our observations are that if a cycle is encountered it is only with a small portion of the pixels. We can halt the iterations if this cycling occurs and the result is sufficient to achieve the desired results.

The general procedure for the non-cooperative game is outlined in Algorithm 3.1. The inputs for the algorithm are the original image, u, some initial segmentation w^0, as well as a limit on the number of iterations, K, which will be a model parameter. The initial segmentation comes from the initialization process described in Section 3.2 or from previous iterations of the iterative procedure described in Section 3.1.4.

Algorithm 3.1 Non-Cooperative Segmentation

Input: Image u, initial segmentation w^0
Output: Segmentation $w^{non-coop}$, local energies $E^{non-coop}$

```
 1:
 2: function Non-Cooperative-Segmentation($u, w^0$)
 3:     for $i = 1$ to $K$ do
 4:         Compute $\mu_0, \sigma_0, \mu_1, \sigma_1$ from $w^{i-1}$ and $u$
 5:         for Each pixel $s \in S$ do
 6:             Compute $E^0 = E^0(u, w^{i-1}, s)$
 7:             Compute $E^1 = E^1(u, w^{i-1}, s)$
 8:             if $E^0 < E^1$ then
 9:                 Set $w_s^i = 0$
10:                 Set $E_s^{non-coop} = E^0$
11:             else
12:                 Set $w_s^i = 1$
13:                 Set $E_s^{non-coop} = E^1$
14:             end if
15:         end for
16:         if $w^i = w^{i-1}$ then
17:             return $w^i, E^{non-coop}$
18:         end if
19:     end for
20:     return $w^K, E^{non-coop}$
21: end function
```

Each iteration i of the algorithm starts by computing the mean and standard deviation for the two classes 0 and 1. This is done by using the segmentation labels provided by the previous iteration (or the initial segmentation w^0). The algorithm proceeds by computing energies E^0 and E^1 for each pixel s using equations (3.10) and (3.11). Finally the algorithm chooses the preferred label w_s^i for each pixel by comparing E^0 and E^1. It also stores the value of the local energy it has chosen in the variable $E^{non-coop}$, which is also indexed by S. These local energies and the final segmentation $w^{non-coop}$ are the outputs of the algorithm.

Segmentation results from this algorithm are generally not very good. In this non-cooperative game the agents make decisions base only on information from their four neighbouring pixels. This tends to produce a lot of noise in the final segmentation. We wish to provide a way for the pixels to incorporate information into their decisions from further away in the image. In the next section we introduce a cooperative component to the game that allows the agents to do just this.

3.1.3 Cooperative Strategy

To introduce cooperation into the game that the agents are playing we first introduce the concept of a coalition. Here we define a coalition C to be a collection of pixels $C \subseteq S$, such that all pixels are all spatially connected, meaning all pixels can be reached from one another by travelling amongst shared neighbours. We can split the entire image into coalitions in this way. We introduce the collection of all coalitions, $\mathcal{C} = \{C_1, C_2, \ldots, C_P\}$, with the properties that the entire image is divided amongst the coalitions, $S = C_1 \cup C_2 \cup \ldots \cup C_P$, and that no two coalitions share any pixels, $C_i \cap C_j = \emptyset, \forall i \neq j$.

We also introduce an extension to the local energy functions from (3.10) and (3.11) to define energies for an entire coalition:

$$E^0(u, w, C) \equiv \sum_{s \in C} E^0(u, w^{C,0}, s), \tag{3.12}$$

$$E^1(u, w, C) \equiv \sum_{s \in C} E^1(u, w^{C,1}, s). \tag{3.13}$$

Here we use special segmentations $w^{C,0}$ and $w^{C,1}$ as input to the local energy functions; $w^{C,0}$ is a modification of the segmentation w in which every pixel $s \in C$ is re-labelled 0, and similarly $w^{C,1}$ is a modification of the segmentation w in which every pixel $s \in C$ is re-labelled 1.

With a set of coalitions, $\mathcal{C}$, we proceed by allowing each coalition to decide as a group which strategy all pixels belonging to that coalition will take. This procedure is detailed in Algorithm 3.2. Inputs to the algorithm are the image u, an initial segmentation w^0, and a set of coalitions $\mathcal{C}$.

Algorithm 3.2 Cooperative Segmentation

```
Input: Image $u$, initial segmentation $w^0$, coalitions $\mathcal{C}$
Output: Segmentation $w^{coop}$, local energies $E^{coop}$
 1: function COOPERATIVE-SEGMENTATION($u, w^0, \mathcal{C}$)
 2:     Compute $\mu_0, \sigma_0, \mu_1, \sigma_1$ from $w^0$ and $u$
 3:     for $C \in \mathcal{C}$ do
 4:         Let $w^{C,0} = w^{C,1} = w$
 5:         Set $w_s^{C,0} = 0, \forall s \in C$
 6:         Set $w_s^{C,1} = 1, \forall s \in C$
 7:         Compute $E^0 = \sum_{s \in C} E^0(u, w^{C,0}, s)$
 8:         Compute $E^1 = \sum_{s \in C} E^1(u, w^{C,1}, s)$
 9:         if $E^0 < E^1$ then
10:             $\forall s \in C$, set $w_s^{coop} = 0$
11:             $\forall s \in C$, set $E_s^{coop} = E^0(u, w^{C,0}, s)$
12:         else
13:             $\forall s \in C$, set $w_s^{coop} = 1$
14:             $\forall s \in C$, set $E_s^{coop} = E^1(u, w^{C,1}, s)$
15:         end if
16:     end for
17:     return $w^{coop}, E^{coop}$
18: end function
```

The algorithm begins by computing the means and standard deviations for the classes 0 and 1 specified by the initial segmentation w^0. It continues by computing coalition energies for each $C \in \mathcal{C}$. It computes the energy E^0 given that the entire coalition C takes label 0 using equation (3.12) and the energy E^1 given that the entire coalition C takes label 1 using equation (3.13). By comparing these two coalition wide energies, the coalition makes a decision as to which label all its members should take. It stores the relevant local energies and also the labels that are chosen by each coalition. We are left at the end with the cooperative segmentation w^{coop} and the collection of local energies, E^{coop}, which each pixel computed when choosing its label.

This cooperative game tends to produce large coalitions, and has trouble segmenting fine details in the image because there is no way to reduce coalition size. To achieve a segmentation with strengths from both the the non-cooperative routine as well as this cooperative one, we incorporate the two into a single segmentation algorithm, described in the next section.

3.1.4 Strategy Integration

In this section we describe the procedure by which the non-cooperative and cooperative segmentation routines are integrated into a single procedure. We have segmentations produced by those routines: $w^{non-coop}$ and w^{coop}. We also have the collection of local energies computed as each pixel chose its label in Algorithms 3.1 and 3.2: $E^{non-coop}$ and E^{coop}.

Normally two segmentations would be hard to compare and combine, however we built each routine upon the same local energy functions (3.10) and (3.11). To produce a combined segmentation, w^{comb}, we simply compare the cooperative and non-cooperative energies from $E^{non-coop}$ and E^{coop}. The details of this are presented in Algorithm 3.3.

Algorithm 3.3 Combine Non-Cooperative and Cooperative Segmentations

```
Input: Segmentations $w^{non-coop}$ and $w^{coop}$, energies $E^{non-coop}$ and $E^{coop}$
Output: Segmentation $w^{comb}$
 1: function COMBINED-SEGMENTATION($w^{non-coop}, w^{coop}, E^{non-coop}, E^{coop}$)
 2:     for $s \in S$ do
 3:         if $E_s^{non-coop} < E_s^{coop}$ then
 4:             Set $w_s^{comb} = w_s^{non-coop}$
 5:         else
 6:             Set $w_s^{comb} = w_s^{coop}$
 7:         end if
 8:     end for
 9:     return $w^{comb}$
10: end function
```

As mentioned, the inputs to the algorithm are the two segmentations and the two collections of local energies: $w^{non-coop}$, w^{coop}, $E^{non-coop}$, and E^{coop}. For each pixel s, the algorithm compares the local energies $E_s^{non-coop}$ and E_s^{coop}. Whichever has the smaller energy, provides the label for the combined segmentation w^{comb}. In this way each pixel in the combined segmentation is provided with a label from wither $w^{non-coop}$ or w^{coop}.

Similar to the iterations in the non-cooperative component with the intent of converging to a Nash equilibrium, we can iterate this combination procedure until a stable segmentation is reached. At each step we must update the coalitions with information from the new combined segmentation. To form the new set of coalitions we group all spatially connected pixels with the same label in w^{comb}. In this work we consider only maximal coalitions. That is coalitions of spatially connected pixels taking the same strategy must be in the same coalition. This can be done using breadth first search, the details of which are detailed in Algorithm 3.4.

Algorithm 3.4 Coalition Reformation

```
Input: Segmentation $w$
Output: Set of coalitions $\mathcal{C}$
 1: function GET-COALITIONS($w$)
 2:     Initialize $C_1 = \{\}$
 3:     $i = 1$
 4:     Initialize $\mathcal{C} = \{\}$
 5:     Initialize $M_s = false, \forall s \in S$
 6:     while $\exists s$, such that $M_s = false$ do
 7:         Find $s$, such that $M_s = false$
 8:         Let $Q = \{s\}$
 9:         while $Q$ not empty do
10:             Let $Q = Q \setminus \{s\}$
11:             Let $C_i = C_i \cup \{s\}$
12:             Set $M_s = true$
13:             for $r \in N(s)$ do
14:                 if $w_s = w_r$ and $M_r = false$ then
15:                     Let $Q = Q \cup \{r\}$
16:                 end if
17:             end for
18:         end while
19:         Let $\mathcal{C} = \mathcal{C} \cup C_i$
20:         Let $i = i + 1$
21:         Let $C_i = \{\}$
22:     end while
23:     return $\mathcal{C}$
24: end function
```

The breadth first search marks each pixel s as it is found, storing that information in M. The algorithm continues as long as there are unmarked pixels. When an unmarked pixel s is found, a new coalition is formed. The algorithm proceeds by adding all pixels r spatially connected to s that have the same label ($w_s = w_r$) to the coalition. All pixels in this coalition are then marked and we proceed to look for another unmarked pixel. The end result is a new set of coalitions $\mathcal{C}$.

With a new set of coalitions we repeat the process of computing both the non-cooperative and cooperative segmentations, as well as performing the combination procedure. These steps are iterated a certain number of times specified by parameter L. In general, the algorithm is found to converge in ~ 10 iterations.

The complete algorithm is presented in Algorithm 3.5. The inputs to the algorithm are an image u, an initial segmentation w^0 and an initial set of coalitions $\mathcal{C}$. The procedure of how to get w^0 and $\mathcal{C}$ are discussed in Sections 3.2 and 3.3.

Algorithm 3.5 Game Theoretic Segmentation

```
Input: Image $u$, initial segmentation $w^0$, initial coalitions $\mathcal{C}$
Output: Segmentation $w^{gt}$
 1: function GT-SEGMENTATION($u, w^0, \mathcal{C}$)
 2:     $w^{gt} = w^0$
 3:     for $i = 1$ to $L$ do
 4:         $w^{non-coop}, E^{non-coop}$ = NON-COOPERATIVE-SEGMENTATION($u, w^{gt}$)
 5:         $w^{coop}, E^{coop}$ = COOPERATIVE-SEGMENTATION($u, w^{gt}, \mathcal{C}$)
 6:         $w^{gt}$ = COMBINED-SEGMENTATION($w^{non-coop}, w^{coop}, E^{non-coop}, E^{coop}$)
 7:         $\mathcal{C}$ = GET-COALITIONS($w^{GT}$)
 8:     end for
 9:     return $w^{gt}$
10: end function
```

Each iteration of the algorithm starts by computing ($w^{non-coop}, E^{non-coop}$) using Algorithm 3.1 and (w^{coop}, E^{coop}) using Algorithm 3.2. We then produce a combined segmentation w^{comb} using Algorithm 3.3. The final step is to recompute the coalitions for use in the next iteration using Algorithm 3.4.

The final game theoretic segmentation w^{gt} produced by this algorithm is found to be quite good in practice. We will apply some post-processing procedures to eliminate noise that is often present when noisy images are used as input. This process is described in Section 3.4. Numerical results and sample segmentations using this algorithm are presented and discussed in Chapter 4.

3.2 Segmentation initialization using two class k-means

The game theoretic segmentation presented in Section 3.1 requires an initial segmentation as input. This could be a circle dividing up the image into two groups or some other trivial division of the pixels. Experiments with several methods of choosing an initial segmentation showed that the algorithm converges more quickly if the initial segmentation has some resemblance to the final segmentation.

We use the k-means algorithm described in Section 2.3 for this purpose. We consider only the two class k-means algorithm ($k = 2$) and our input will be the pixel intensities which are scalars. This gives us a simplified version of equation (2.1). Using the notation from our image segmentation problem we get the k-means optimization problem:

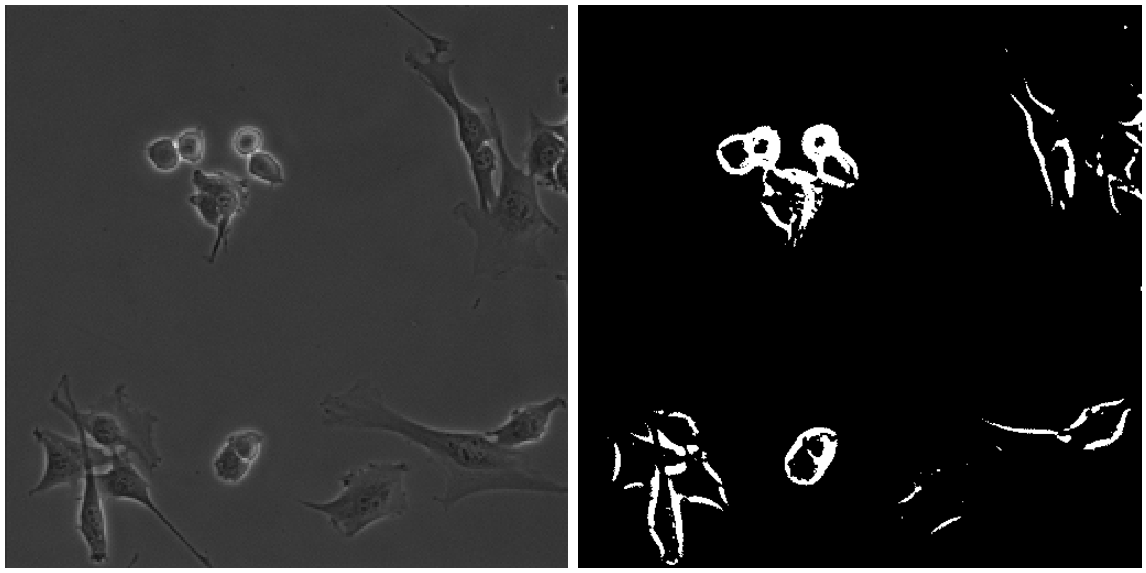
$$\arg\min_{w} \sum_{s:\ w_s=0} \|u_s - \mu_0\|^2 + \sum_{s:\ w_s=1} \|u_s - \mu_1\|^2. \tag{3.14}$$

This greatly simplified k-means runs quickly even for a large number of pixels.

The k-means algorithm will divide the pixels into two groups based on intensity. This is generally not a good segmentation for low contrast images such as the cellular images we are concerned with, but the k-means algorithm for two classes with scalar inputs is fast, and the result is sufficient for our initialization purposes.

An added benefit is that the k-means algorithm implicitly returns the mean of each class which saves that computation from the first iteration of the game theoretic segmentation. K-means is a heuristic algorithm and therefore not guaranteed to converge to the global optimum. This is again is acceptable for our purposes as we require only a rough segmentation to initialize our algorithm.

Figure 3.1 shows an example k-means segmentation used as input to the game theoretic segmentation.



(a) Bright-field cellular image (b) Two class k-means segmentation

Figure 3.1: Two class k-means run on bright-field cellular image pixel intensities.

3.3 Coalition initialization

The cooperative component of the game theoretic segmentation from Section 3.1.3 requires an a initial set of coalitions $\mathcal{C}$. Using Get-Coalitions from Algorithm 3.4 after the k-means segmentation is one option, however there are a few problems that rule it out. The cell images we are segmenting are often noisy. This results in a k-means segmentation that is quite noisy. If we run Get-Coalitions on this segmentation we end up with numerous coalitions that consist of only a few or even single pixels. These numerous small coalitions cause a lot of work in the cooperative component of the algorithm while at the same time losing the more global property we desire from our coalitions.

We instead divide the entire image into square blocks of size $B \times B$, which become our coalitions. B is a model parameter that must be selected carefully; we balance block sizes being too small to give desirable coalition benefits or too large and losing resolution in the cooperative segmentation resulting from the coalitions. On the first iteration of the game theoretic algorithm the coalitions will not represent spatially connected pixels with the same label as they do in the remaining iterations.

Figure 3.2 shows an example cooperative segmentation resulting from these block coalitions.

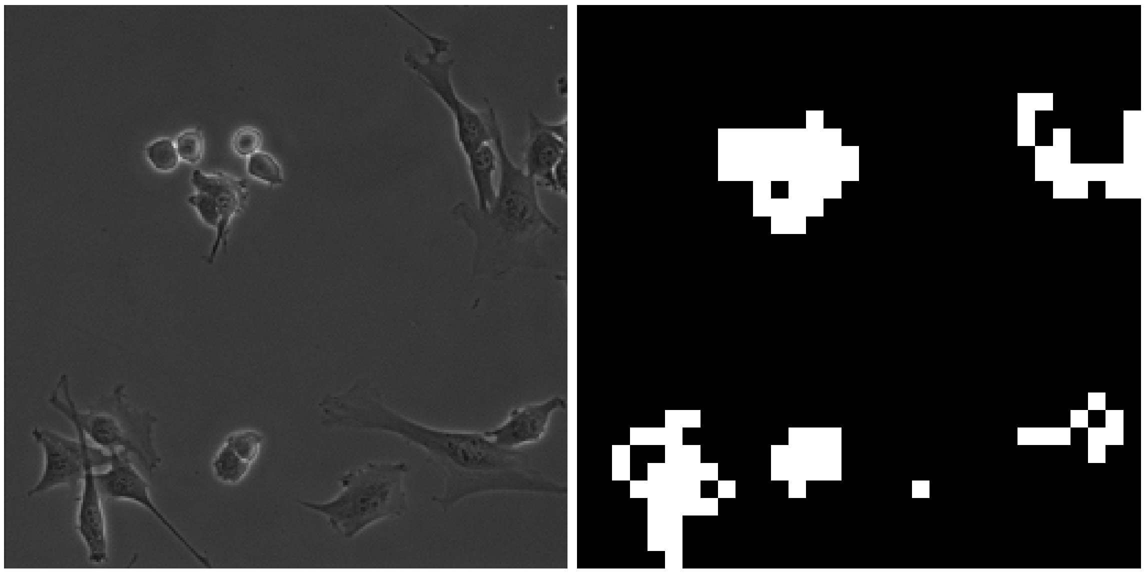

(a) Bright-field cellular image (b) Cooperative segmentation using 8×8 block coalitions

Figure 3.2: First iteration for cooperative segmentation on bright-field cellular image.

3.4 Smoothing using Chan-Vese Active Contours

We use the Chan-Vese active contours method described in Section 2.4 as a post-processing step after the game theoretic segmentation algorithm. The game theoretic algorithm can often have a fair amount of noise in the final segmentation, and we use the active contours method to achieve a smoother segmentation. We use the binary output of our game theoretic segmentation as input to the active contours algorithm.

3.4.1 Initialization

The function ϕ defining our boundary should be a signed distance function. For a general image, Chan-Vese segmentation usually begins with a circle or other simple shape defining the boundary. The signed distance function in this case is easy to define. We however would like to start with the zero level being the boundary defined by the output of the game theoretic segmentation.

Computing the signed distance for the complicated shapes resulting from the image segmentation is challenging. We instead approximate this by computed a signed Manhattan distance on the grid specified by the pixels. This is done by giving interior pixels next to the boundary a value of 1. Interior pixels that are neighbours to the pixels of value 1 are given a value of 2 and so on. The same procedure is done for pixels outside of the boundary however the values are negative. An example of this can be seen in Figure 3.3.

-3	-2	-1	-1	-2	-3	-4
-2	-1	1	1	-1	-2	-3
-1	1	2	2	1	-1	-2
-1	1	2	2	1	-1	-2
-1	1	2	3	2	1	-1
-1	1	2	2	1	-1	-2
-1	1	1	1	-1	-2	-3

Figure 3.3: Example initialization of ϕ with interior pixels (gray) positive and exterior pixels (white) negative.

3.4.2 Smoothing Procedure

The active contours algorithm uses several parameters to control the resulting segmentation: $\rho, \lambda_1, \lambda_2, h, \Delta t, T$. With these parameters specified, and ϕ^0 initialized as in Section 3.4.1, we are ready to begin the algorithm. We solve equation (2.5) for ϕ^{n+1}, and iterate until convergence or until we have reached T iterations. Generally ~ 10 iterations is sufficient for our smoothing purposes.

The final output is a significantly smoother version of the segmentation generated from the game theoretic algorithm. A comparison of before and after smoothing can be seen in Figure 3.4.

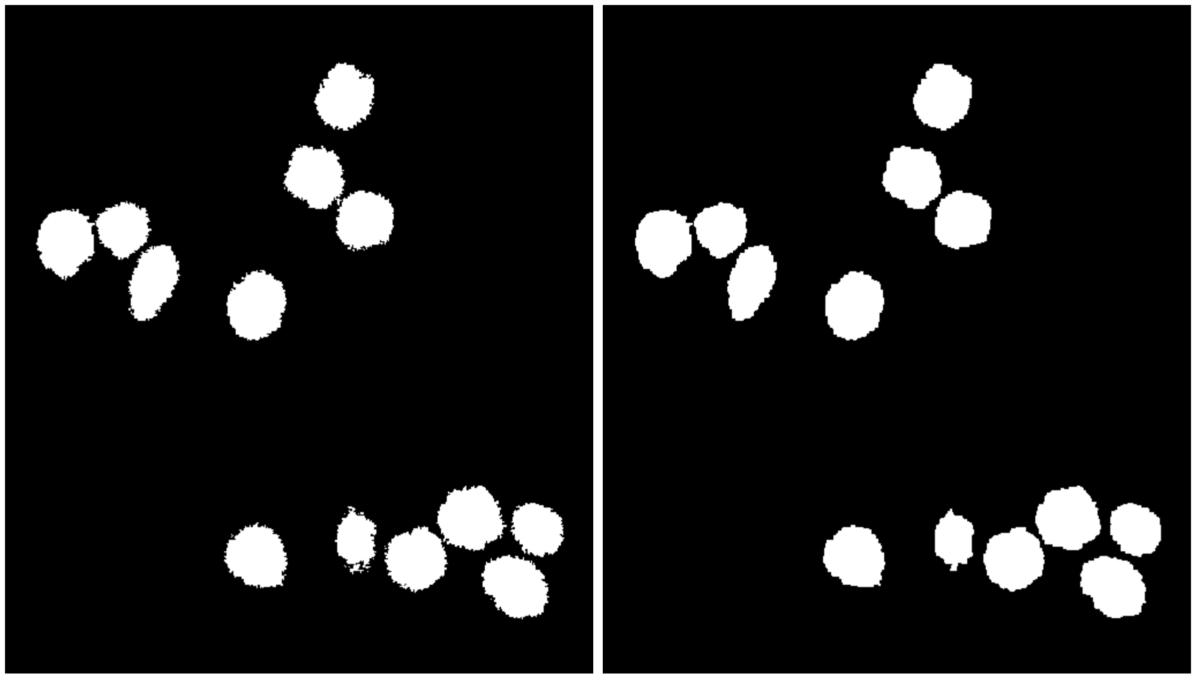

(a) Segmentation before smoothing (b) Segmentation after smoothing

Figure 3.4: Sample output from the smoothing procedure using active-contours.

Chapter 4

Numerical Results

From Chapter 3 we have a collection of parameters we must specify for the model. From the game theoretic portion we have: B, the block size of the initial coalitions; L, the maximum number of iterations we will perform; β, a parameter in the energy function controlling the tendency to allow noise; and K, the number of iterations allowed in the non-cooperative component. In all cases we fix $L = 10$ and $K = 30$.These values were deemed sufficient for convergence in every instance tested.

For the smoothing procedure we use the Chan-Vese active contour algorithm which itself has a number of parameters: h, the space step size; Δt, the time step size; λ_1, λ_2 and ρ, scaling parameters; and T, the maximum number of iterations. In all cases we fix $h = 1$, $\Delta t = 0.3$, λ_1, $\lambda_2 = 1$, and $T = 10$. The values for h, λ_1, and $lambda_2$ are taken from literature and the values for Δ_t and T were selected from experimentation.

When listing parameters for a given segmentation result only B, β, μ will be specified. A small B is often required for images with many small details. The parameter β has the most effect on the resulting segmentation; a small β punishes pixels less for choosing strategies different from their neighbours. Several consequences result from choosing a small β value, we can segment regions of the image that are harder to distinguish from the background but at the same time it can also lead to over segmentation and more noise in the final result. Large β values swing the trade-off in the other direction, giving less noise and less over-segmentation, but less difficult to segment regions are ignored. The parameter ρ controls the smoothness of the result produced by Chan-Vese, in other words how aggressively it will eliminates noise and sharp corners. This parameter can vary depending on the general shape of the cells in the images we wish to segment.

4.1 Bright-Field Images

The bright-field images used are of C2C12 cells [26]. The result of the two segmentation on the same bright-field image can be seen in Figures 4.1 and 4.2. In Figure 4.1 we have chosen a large value for β, while in Figure 4.2 a smaller β. We can see that the algorithm captures most of the cells. Two cells on the right hand side of the image contain some difficult to capture low contrast regions. The result with smaller β captures more of the difficult cells but it also over-segments in several locations. Which result is better may be a matter of contention.

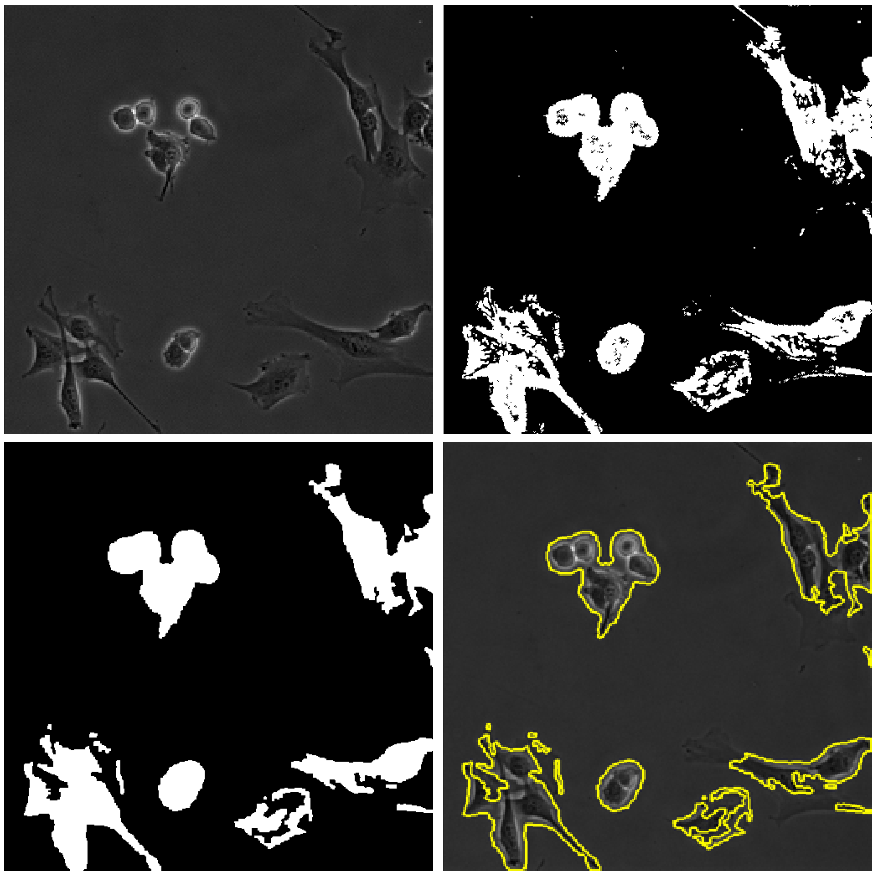


Figure 4.1: Bright-field image segmentation using game theoretic algorithm. Top row: original image, segmentation before smoothing. Bottom Row: final segmentation, visualization $B = 12$, $\beta = 2.2$, $\rho = 6$. Image size: 512×512.

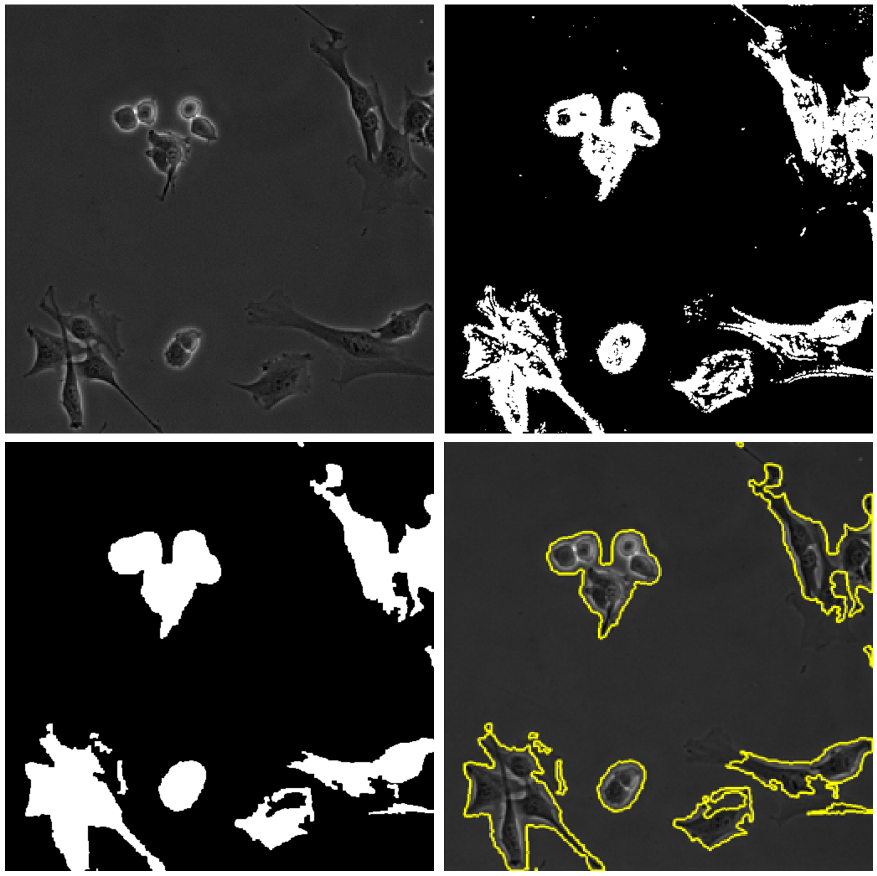


Figure 4.2: Bright-field image segmentation using game theoretic algorithm. Top row: original image, segmentation before smoothing. Bottom Row: final segmentation, visualization. $B = 12$, $\beta = 0.6$, $\rho = 6$. Image size: 512×512.

Improvements can be made if we isolate cells in the image and perform segmentation on them individually. This is not unexpected as the information from other cells could negatively influence the segmentation of a specific cell. Figures 4.3 and 4.4 offer two instance where an isolated segmentation results in an improved result from the full image segmentation.

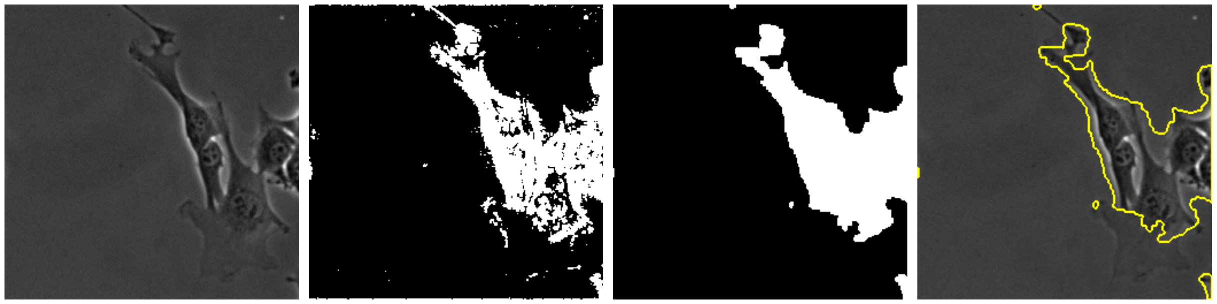


Figure 4.3: Bright-field image segmentation using game theoretic algorithm. Left-to-right: original image, segmentation before smoothing, final segmentation, visualization. $B = 6$, $\beta = 0.5$, $\rho = 6$. Image size: 256×256.

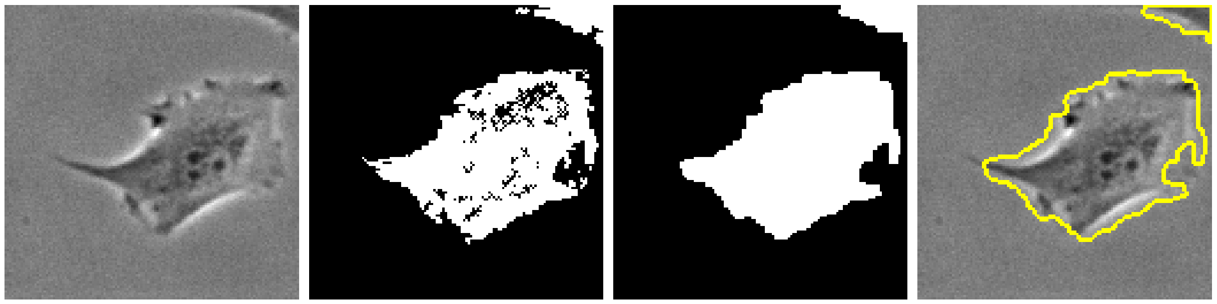


Figure 4.4: Bright-field image segmentation using game theoretic algorithm. Left-to-right: original image, segmentation before smoothing, final segmentation, visualization. $B = 12$, $\beta = 1.2$, $\rho = 6$. Image size: 128×128.

Some timing results for different image size can be found in Table 4.1. Each step up in size is actually a quadrupling in the number of pixels. We see a roughly ten-fold increase in time taken for the game theoretic algorithm at each size increase and less than that for the smoothing portion.

Image Size	Game Theoretic Time (s)	Chan-Vese Time (s)
128×128	1.410101	0.977529
256×256	15.03144	4.95104
512×512	138.67926	34.69564

Table 4.1: Timing results for different size bright-field images

4.2 ISBI 2013 Cell Tracking Challeng Datasets

4.2.1 Jaccard Similarity Index

The results published in [15] use a standard measure to compare segmentation results across the different algorithms. We compare our segmentation results so that we can compare the effectiveness of our algorithms to that of the submitted algorithms described in Section 2.5.2.

From an image frame I, a ground truth G was hand labelled. Due to the number of total cells across all data sets, only a certain number of cells in each frame was labelled, although one frame was guaranteed to be fully labelled in each dataset. For each labelled frame then, we compare it to the segmentation produced by our algorithm. For each labelled cell R, we first decide whether there is a corresponding cell in our segmented image. We consider each cell S present in the segmented image and consider them matching if

$$|R \cap S| > 0.5 \cdot |R| \tag{4.1}$$

The size of a cell or its intersection with another cell is simply the number of pixels in that labelled cell. After we determine that the cells correspond to one another, we measure their similarity using the Jaccard similarity index:

$$J(R,S) = \frac{|R \cap S|}{|R \cup S|} \tag{4.2}$$

We can see that if the segmentation matches the ground truth exactly, $R \cap S = R \cup S$ and so we will have $J(R,S) = 1$. Likewise, if the segmentation does not match the ground truth cell will have $J(R,S) = 0$.

By averaging the Jaccard index across every cell in each frame of a dataset we can produce a score between 0 and 1. This give an indication of how successful the segmentation algorithm is and can be used to compare the effectiveness of different algorithms.

4.2.2 Results

Segmentation results from the IEEE International Symposium on Biomedical Imaging 2013 Cell Tracking Challenge compared with our game theoretic algorithm is presented in Table 4.2.

Dataset	C2DL-MSC		N2DH-GOWT1		N2DL-HeLa		N2DH-SIM					
Series	01	02	01	02	01	02	01	02	03	04	05	06
COM-US	0.06	0.15	0.69	0.39	0.33	0.3	0.75	0.77	0.65	0.64	0.64	0.75
HEID-GE	0.34	0.43	0.84	0.89	0.83	0.8	0.91	0.9	0.82	0.83	0.81	0.89
KTH-SE	0.49	0.66	**0.86**	0.91	**0.88**	**0.89**	0.91	0.89	**0.84**	0.83	0.8	0.83
LEID-NL	N/A	N/A	0.84	0.88	0.74	N/A	**0.93**	0.91	0.83	0.84	0.84	**0.91**
PRAG-CZ	0.15	0.21	**0.86**	0.92	0.78	0.76	0.8	0.8	0.75	0.78	0.76	0.79
UPM-ES	0.41	0.19	0.55	0.63	0.63	0.65	0.91	0.89	0.81	0.82	0.77	0.86
GameTheoretic	**0.55**	**0.75**	**0.86**	**0.93**	0.69	0.73	0.9	**0.92**	**0.84**	**0.86**	**0.94**	0.89

Table 4.2: Segmentation scores for ISBI 2013 Cell Tracking Challenge as well as game theoretic algorithm[1]. Bolded entries are the highest score for that dataset.

There are a number of notable results we can see from the scores. The KTH-SE submission is one the strongest of the submitted algorithms. It performs the best on all but one of the real videos and quite well on the simulated videos. The game theoretic algorithm achieves a better segmentation score than the KTH-SE algorithm on nine of the twelve datasets. The LEID-NL submission did not perform well on the real datasets, but had the best segmentation for five of the six simulated videos. Comparing our game theoretic algorithm to LEID-NL, we can see that it out performs it on five of the six simulated videos. From these comparisons we can conclude that the game theoretic algorithm we have developed is a good segmentation algorithm, which has out performed two of the strongest submissions to the ISBI Cell Tracking Challenge.

Of all the datasets from the challenge, the game theoretic algorithm struggled most with the N2DL-HeLa videos, scoring below several of the submitted algorithms. Sample images and segmentations can be seen in Figures 4.9 and 4.10. We can see from these images that they contain a large number of densely packed cells. We note in particular that the game theoretic algorithm has a tendency to over segment (labelling extra background pixels as belonging to cells near the borders). The over segmentation is especially harmful when there are so many densely packed, as the segmentations of neighbouring cells will begin to overlap. This problem might be addressed with more intensive parameter tuning or might involve a more sophisticated approach to creating coalitions.

The C2DL-MSC datasets provide a different story however. Here we can see big im-

[1]Adapted from http://www.codesolorzano.com/celltrackingchallenge/Cell_Tracking_Challenge/Results_First_CTC.html

provements in the segmentation score by the game theoretic algorithm over those of the submitted algorithms. Many of the submitted algorithms had difficulty segmenting the images from both C2DL-MSC series. The C2DL-MSC datasets are challenging because they contain whole cells with irregular shapes, whereas many of the other datasets contain more uniform rounded cell nuclei. Figures 4.5 and 4.6 show sample images and segmentations for these datasets.

Sample segmentations for the N2DH-GOWT1 series can be seen in Figures 4.7 and 4.8. Of note for this dataset is the large difference in intensities of certain cells. In Figure 4.7 we see a cell near the bottom of the image that was present in the ground truth but not completely segmented by our algorithm. Elsewhere in the image we find segmented cells that are hard to see even with the human eye.

Sample segmentations for the simulated N2DH-SIM series can be seen in Figures 4.11 to 4.16. Of note in these datasets are the bright spots in the cells of Figures 4.15 and 4.16, which lead to low contrast near the borders of the cells.

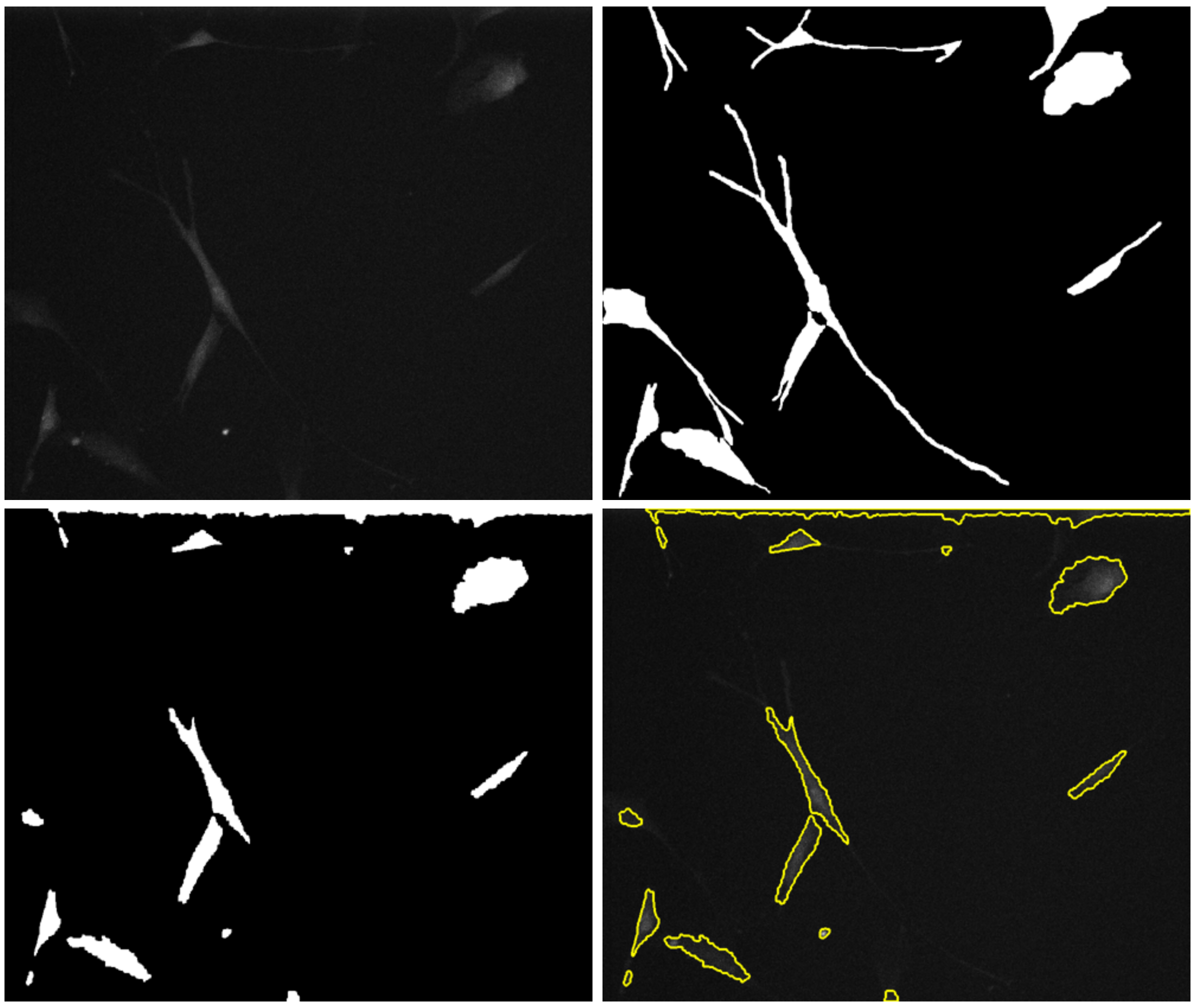


Figure 4.5: Segmentation for C2DL-MSC dataset, series 01. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 16$, $\beta = 2.5$, $\rho = 10$.

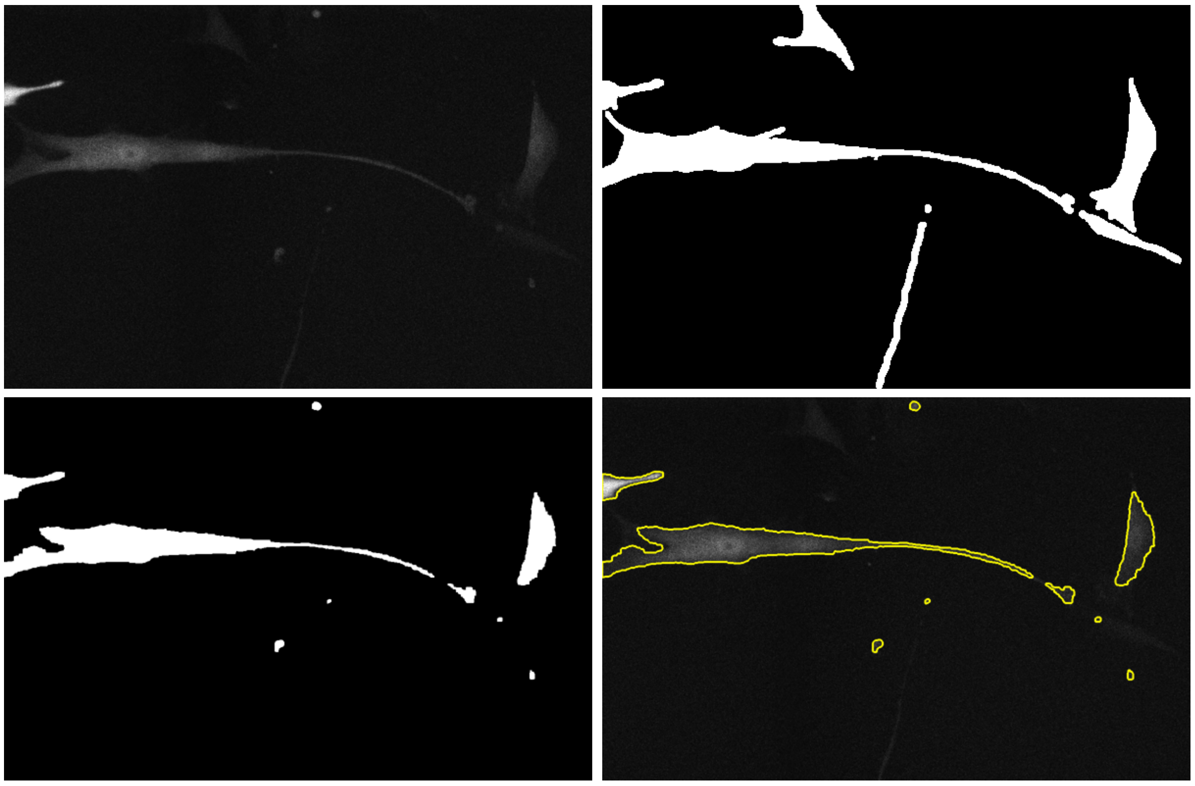


Figure 4.6: Segmentation for C2DL-MSC dataset, series 02. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 16$, $\beta = 2.5$, $\rho = 10$.

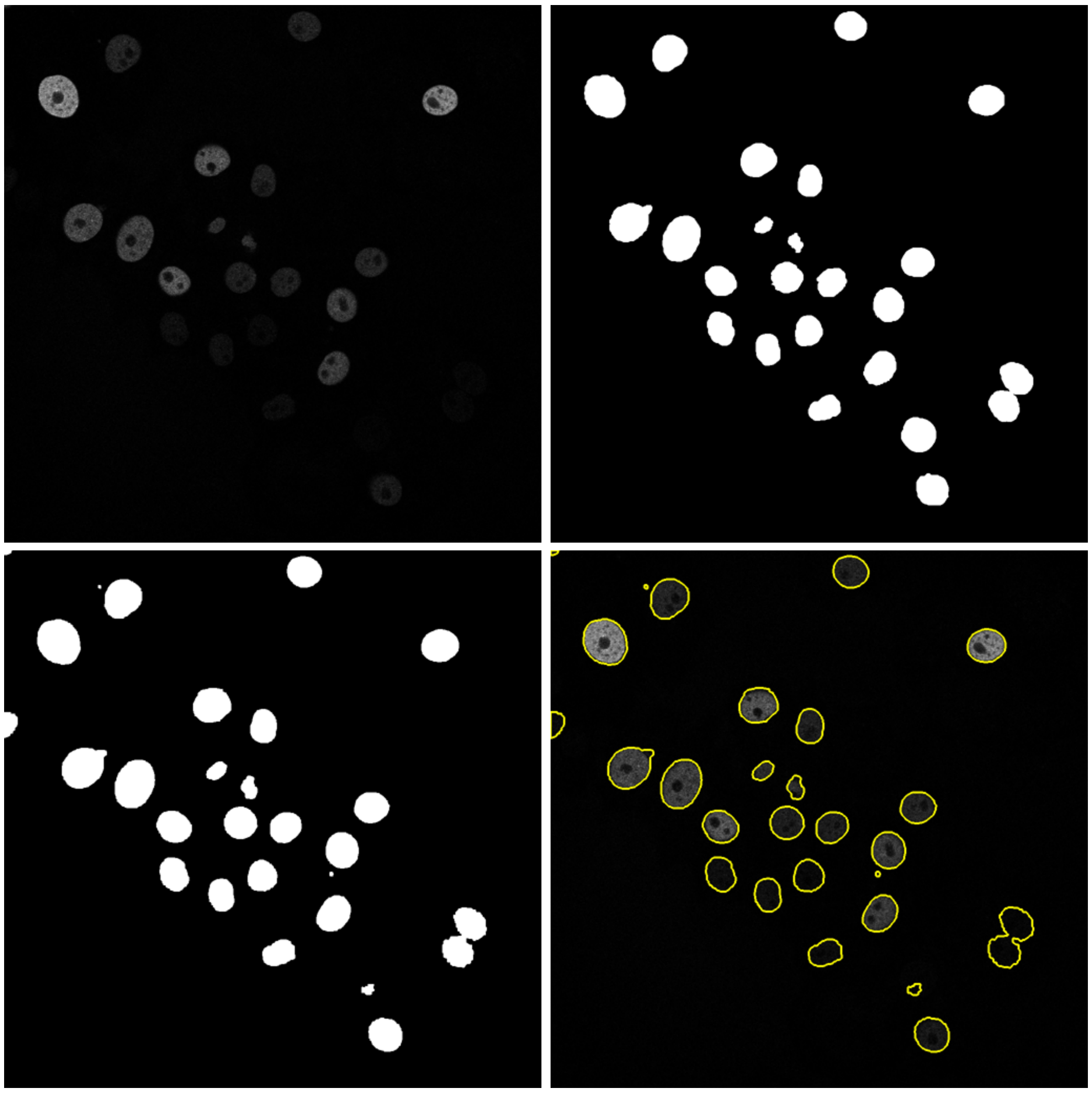


Figure 4.7: Segmentation for N2DH-GOWT1 dataset, series 01. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 25$, $\beta = 2.6$, $\rho = 10$.

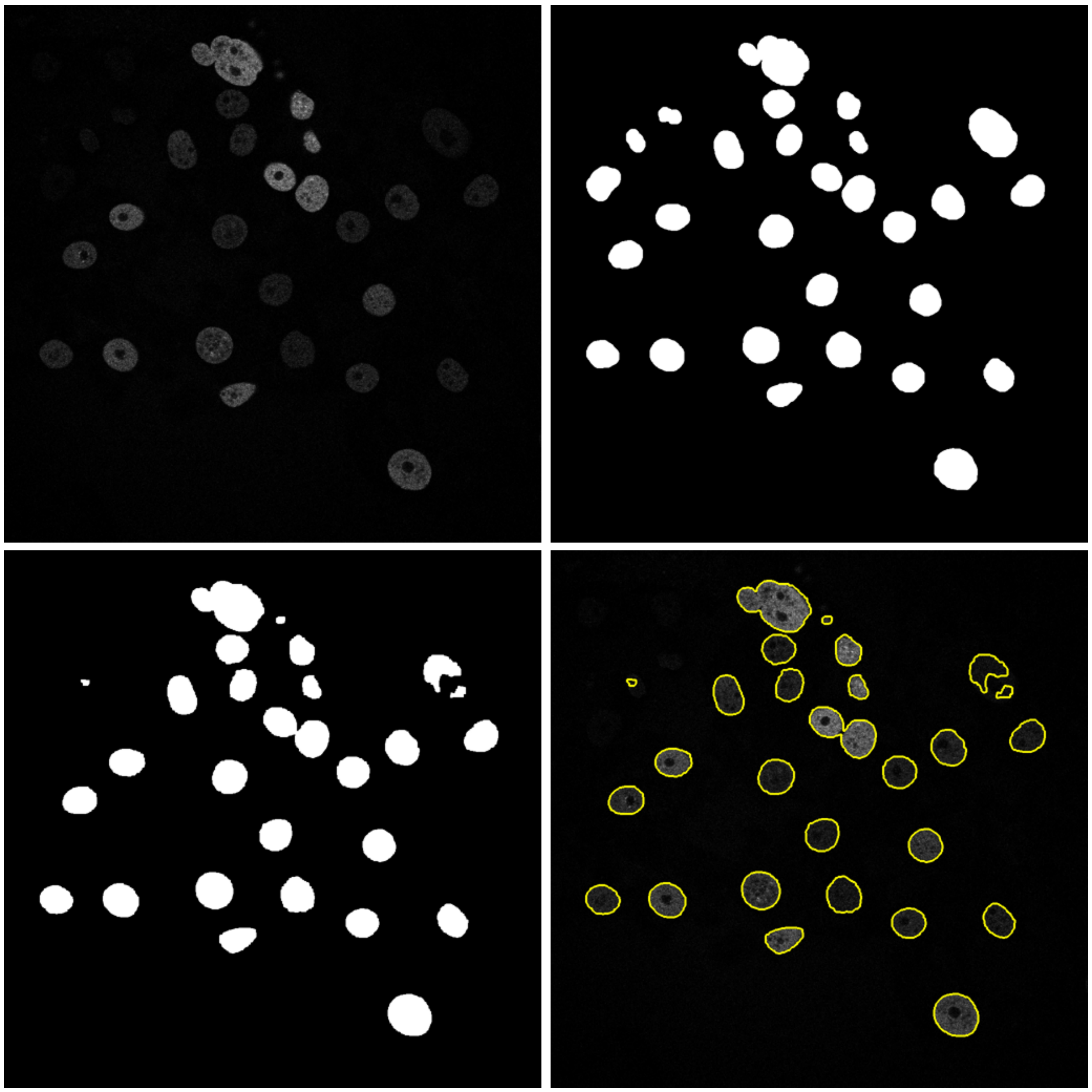


Figure 4.8: Segmentation for N2DH-GOWT1 dataset, series 02. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 16$, $\beta = 1.8$, $\rho = 10$.

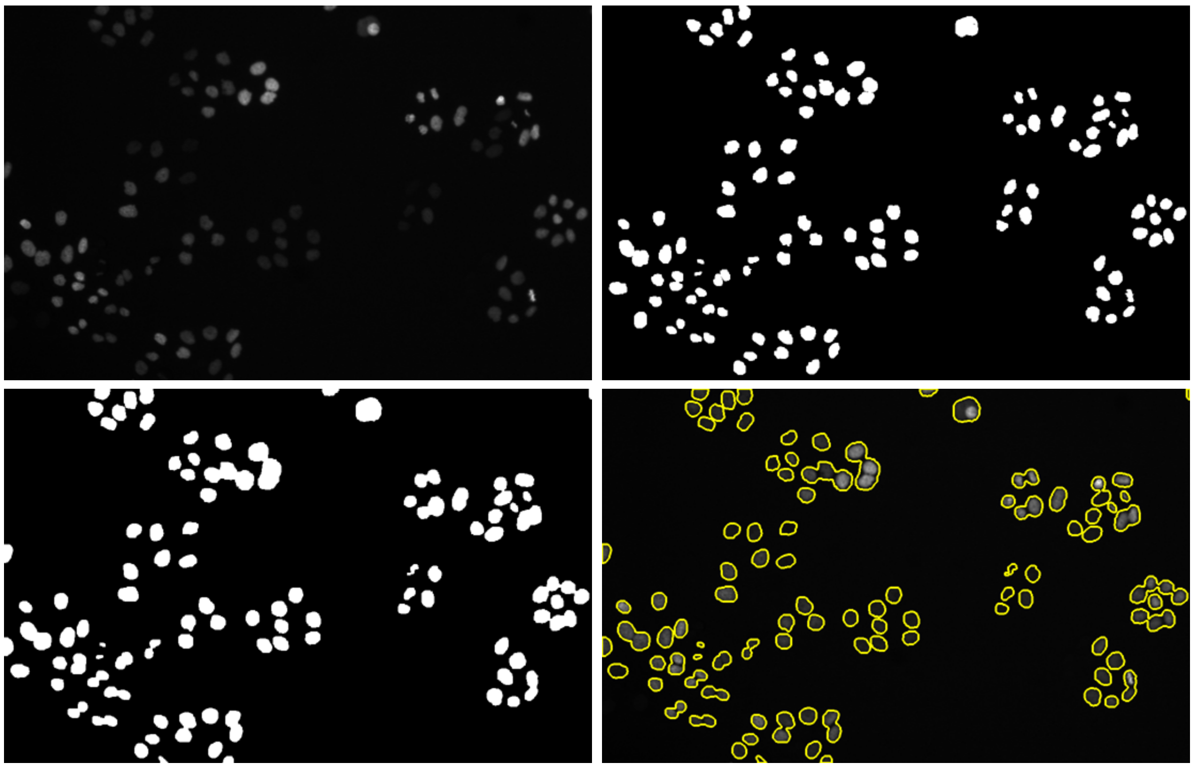


Figure 4.9: Segmentation for N2DL-HeLa dataset, series 01. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 14$, $\beta = 2.2$, $\rho = 10$.

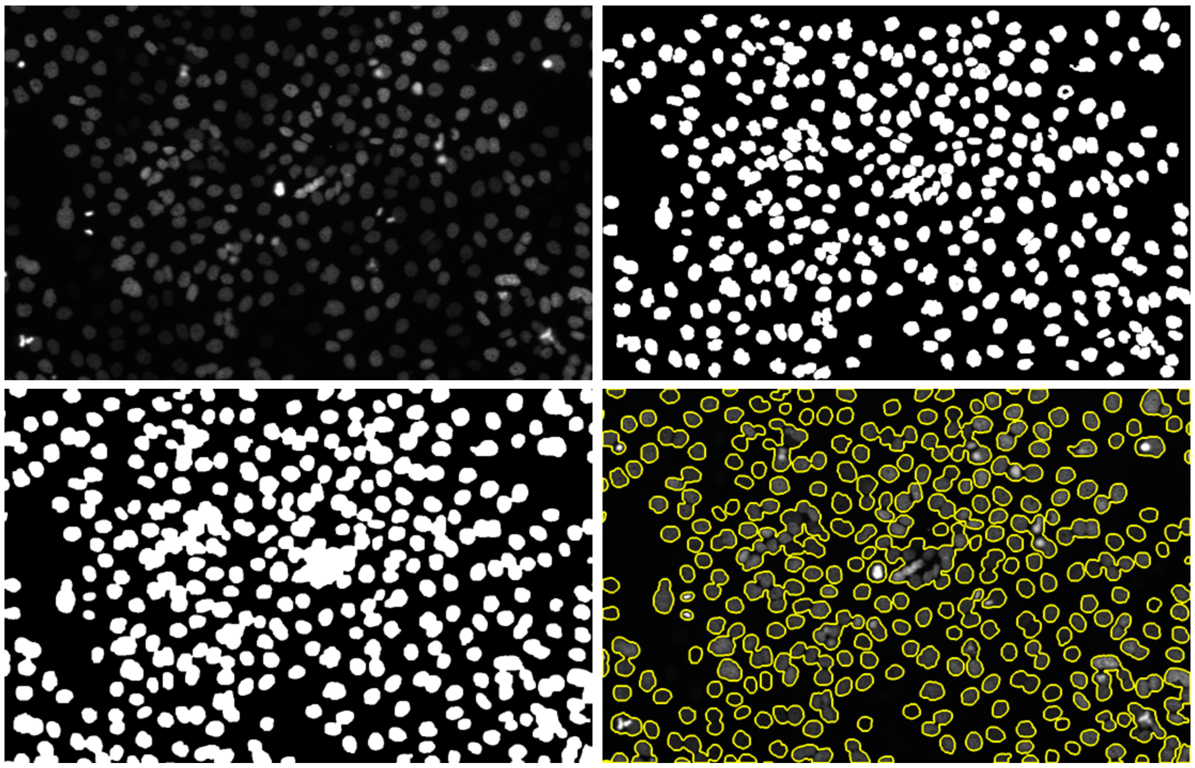


Figure 4.10: Segmentation for N2DL-HeLa dataset, series 02. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 20$, $\beta = 2.0$, $\rho = 10$.

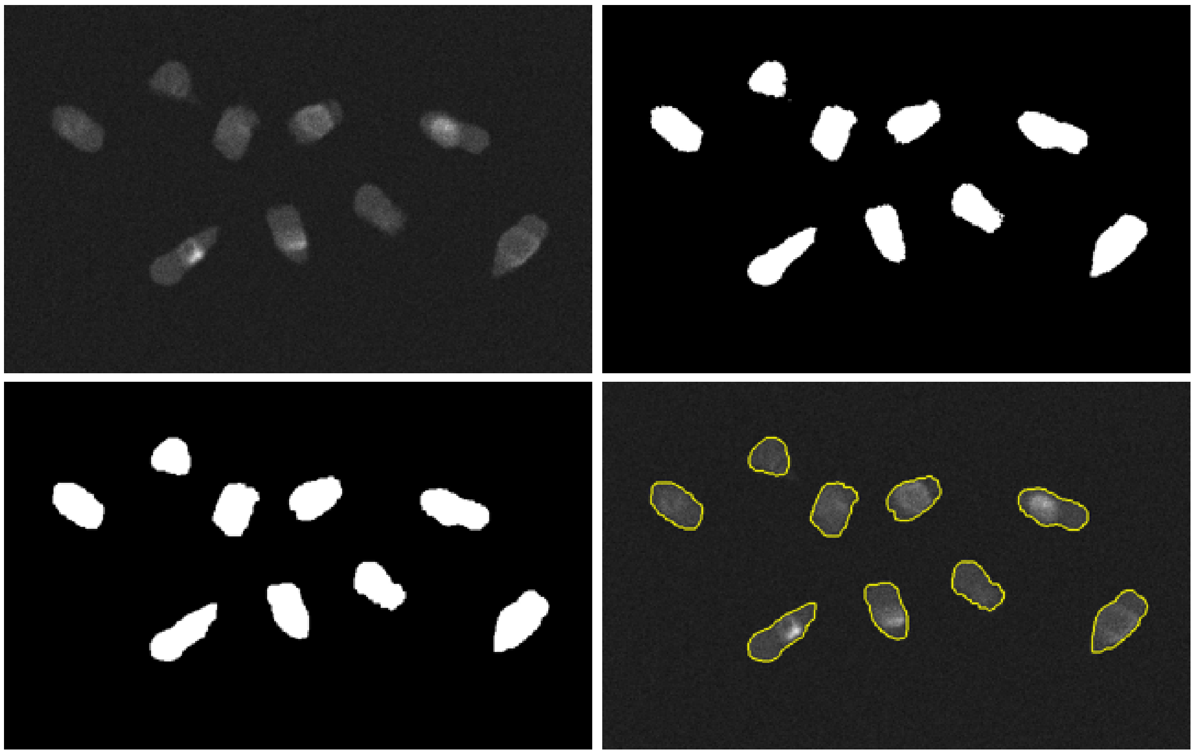


Figure 4.11: Segmentation for N2DH-SIM dataset, series 01. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 20$, $\beta = 1.2$, $\rho = 10$.

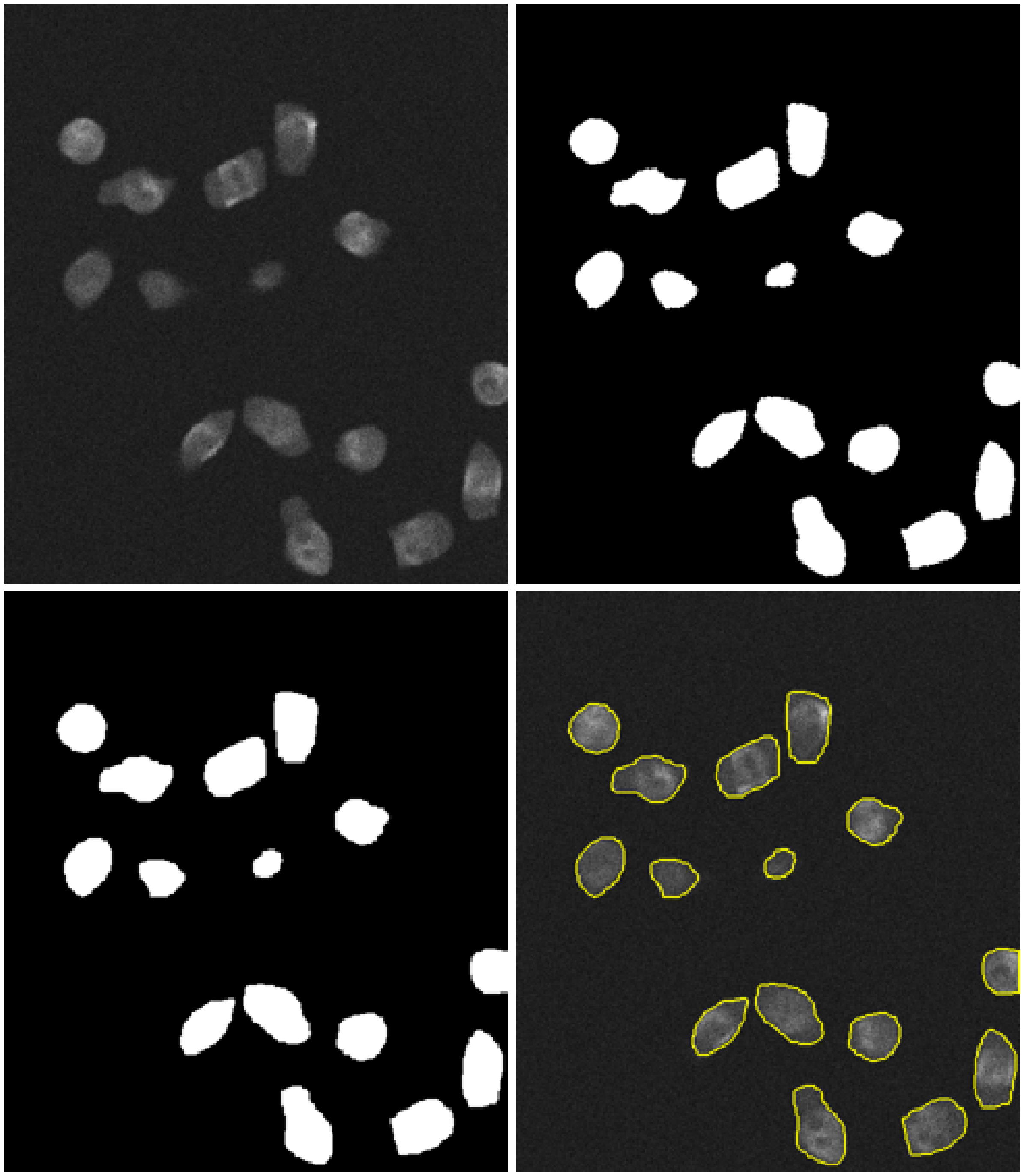


Figure 4.12: Segmentation for N2DH-SIM dataset, series 02. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 20$, $\beta = 1.2$, $\rho = 10$.

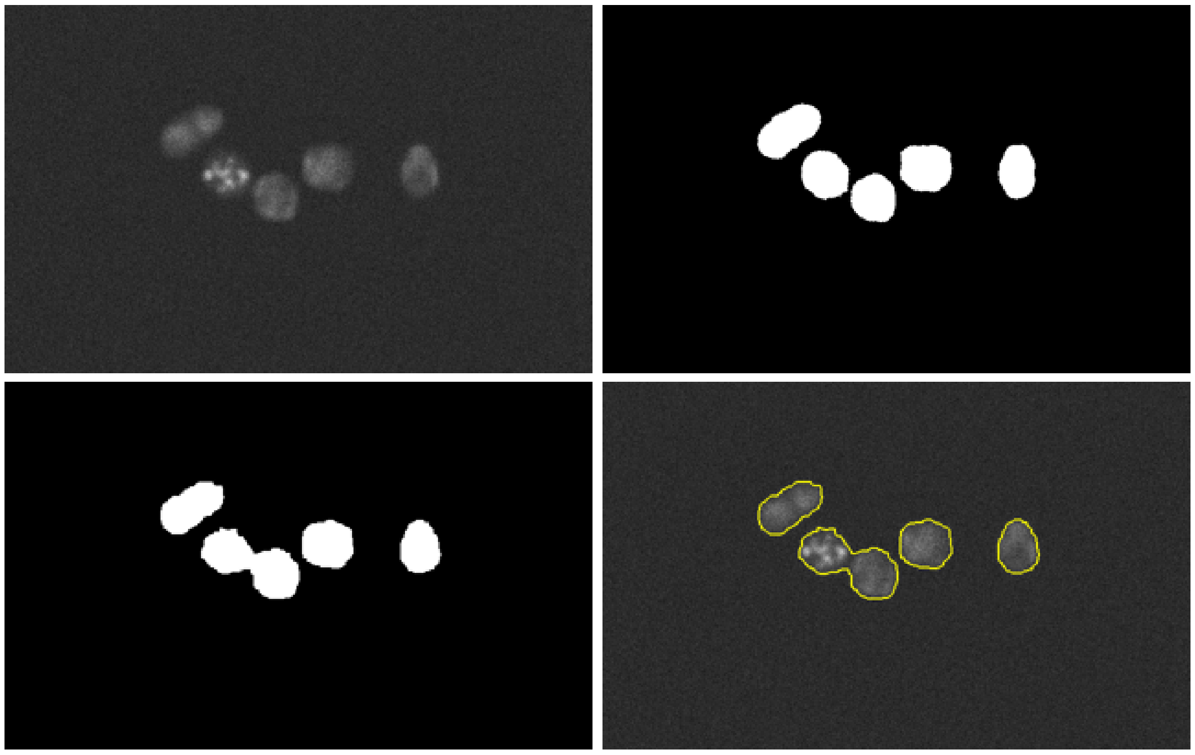


Figure 4.13: Segmentation for N2DH-SIM dataset, series 03. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 12$, $\beta = 1.2$, $\rho = 10$.

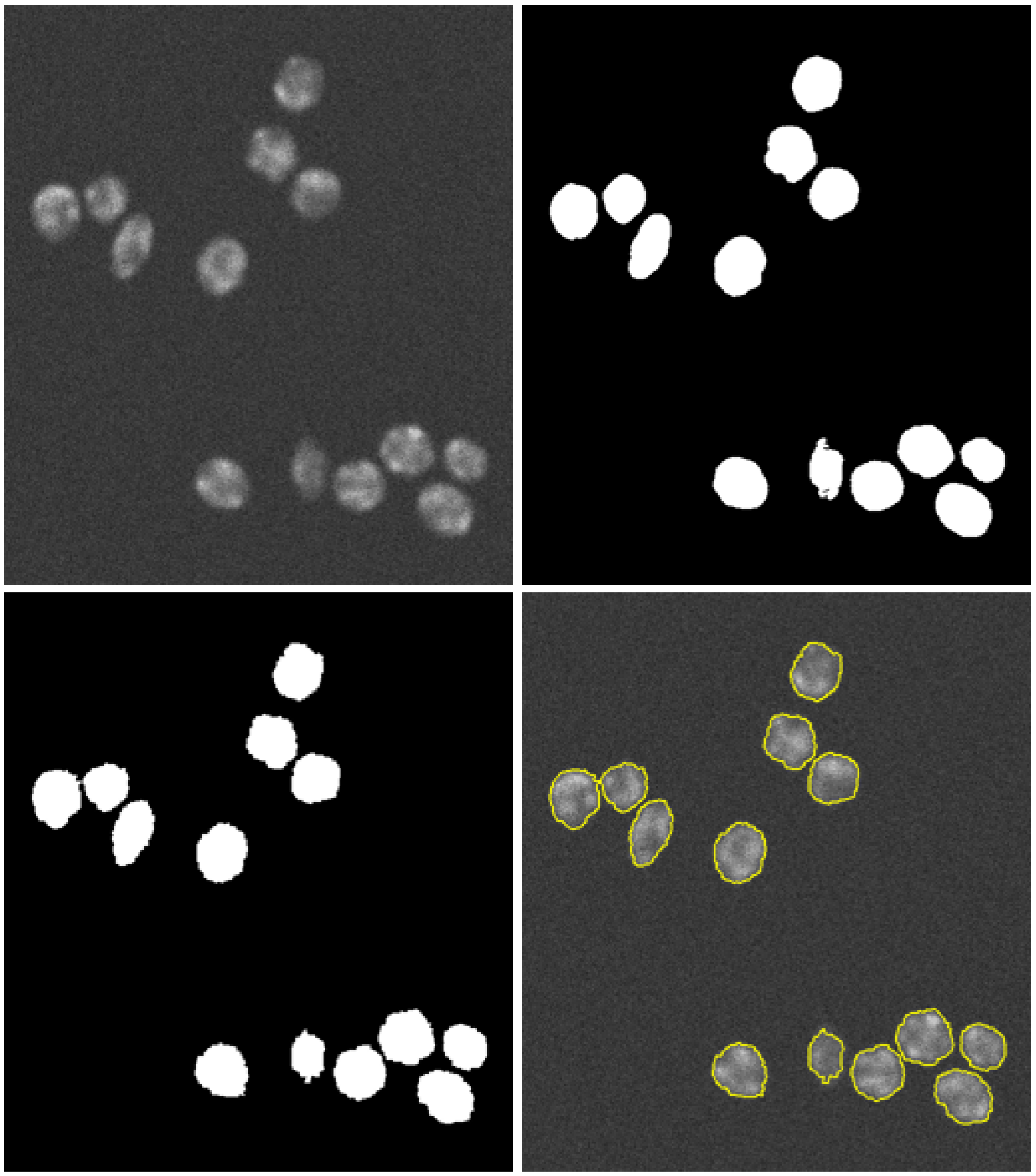


Figure 4.14: Segmentation for N2DH-SIM dataset, series 04. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 16$, $\beta = 1.2$, $\rho = 10$.

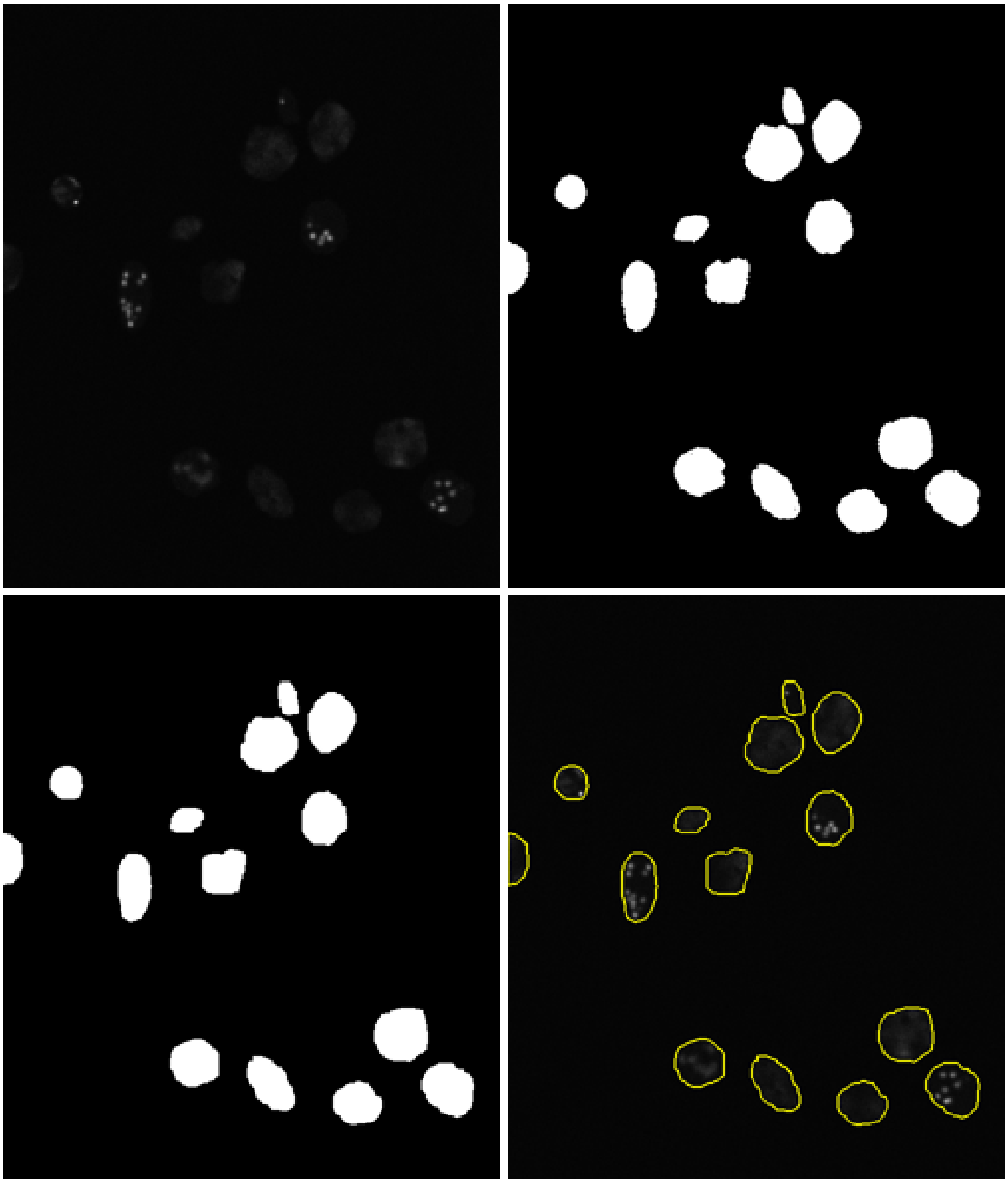


Figure 4.15: Segmentation for N2DH-SIM dataset, series 05. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 25$, $\beta = 1.2$, $\rho = 10$.

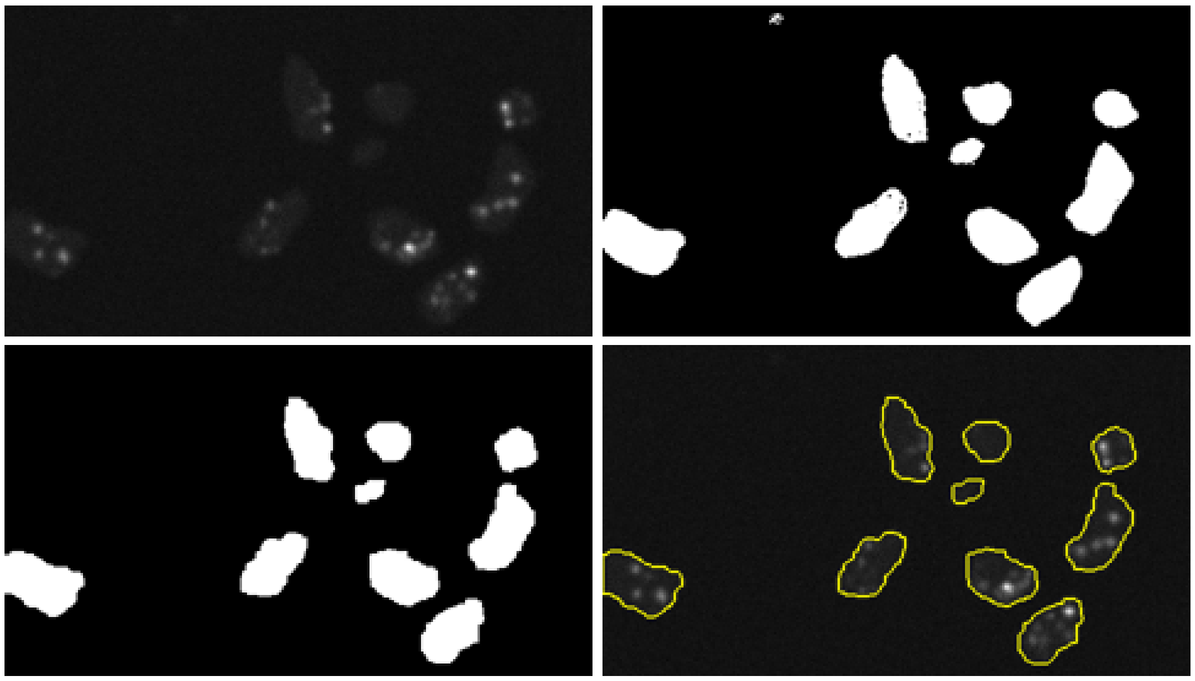


Figure 4.16: Segmentation for N2DH-SIM dataset, series 06. Top row: original image, ground truth. Bottom Row: final segmentation, visualization. $B = 25$, $\beta = 1.2$, $\rho = 10$.

Chapter 5

Conclusions

In this work we developed a novel algorithm to perform segmentation using a combination of game theoretic models not often used for segmentation purposes and more classical segmentation techniques. The game theoretic component combines two paradigms of game theory, cooperative and non-cooperative games, into a single algorithm. Combined with automatic initialization using k-means and post-processing using active contours, the result was a robust segmentation algorithm that requires minimal parameter tuning.

The algorithm was tested using a set of bright-field images as well as twelve datasets from the ISBI 2013 Cell Tracking Challenge. The cell tracking challenge provided twelve 2-dimensional datasets, six of which were simulated and six of which were microscope images. On the bright-field images, the algorithm performed well. The method is able to segment the entire frame containing multiple cells. Results were better when the cells are isolated. This suggests that a more localized approach may produce better results when segmenting the entire frame. For the datasets from the Cell Tracking Challenge, segmentations were compared using the Jaccard similarity index. This allowed a direct comparison between our game theoretic algorithm and the segmentation scores of the algorithms submitted to the competition. Our algorithm perform well across most datasets in the competition. The game theoretic algorithm performed especially well on the challenge dataset C2DL-MSC, which has low contrast and non-uniform cell shapes. This type of image is most similar to the bright-field images with which we compared, and the most difficult to segment of the 2-dimensional datasets provided by the competition.

There are certain areas which might be investigated for future work. The algorithm in general tends to slightly over-segment which causes problems on images such as the N2DL-

HeLa dataset from the cell tracking challenge. This dataset has a very high cell density and the over-segmentation here leads to poor results. Another area of investigation would be in the segmentation of 3-dimensional videos. In this setting the algorithm would be segmenting time-series of 3-dimensional frames. The cell tracking challenge offers datasets of this nature and these types of videos are becoming more common in cellular imaging.

References

[1] R. Adams and L. Bischof, *Seeded region growing*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16 (1994), pp. 641–647.

[2] L. Bradbury and J. W. L. Wan, *A spectral k-means approach to bright-field cell image segmentation*, in 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE Engineering in Medicine and Biology Society Conference Proceedings, 2010, pp. 4748–4751. 32nd Annual International Conference of the IEEE Engineering-in-Medicine-and-Biology-Society (EMBC 10), Buenos Aires, Argentina, Aug 30-Sep 04, 2010.

[3] V. Caselles, F. Catte, T. Coll, and F. Dibos, *A geometric model for active contours in image-processing*, Numerische Mathematik, 66 (1993), pp. 1–31.

[4] A. Chakraborty and J. S. Duncan, *Game-theoretic integration for image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 21 (1999), pp. 12–30.

[5] T. F. Chan and L. A. Vese, *Active contours without edges*, IEEE Transactions on Image Processing, 10 (2001), pp. 266–277.

[6] S. Coraluppi and C. Carthel, *Modified scoring in multiple-hypothesis tracking*, Journal of Advances in Information Fusion, 7 (2012), pp. 153–164.

[7] O. Dzyubachyk, W. A. van Cappellen, J. Essers, W. J. Niessen, and E. Meijering, *Advanced level-set-based vell tracking in time-lapse fluorescence microscopy*, IEEE Transactions on Medical Imaging, 29 (2010), pp. 852–867.

[8] G. Guo, S. Yu, and S. Ma, *An image labeling algorithm based on cooperative game theory*, Signal Processing Proceedings, 1998, 2 (1998), pp. 978–981.

[9] N. Harder, F. Mora-Bermudez, W. J. Godinez, A. Wuensche, R. Eils, J. Ellenberg, and K. Rohr, *Automatic analysis of dividing cells in live cell movies to detect mitotic delays and correlate phenotypes in time*, Genome Research, 19 (2009), pp. 2113–2124.

[10] S. L. Horowitz and T. Pavlidis, *Picture segmentation by a directed split-and-merge procedure*, Proceedings of the 2nd International Joint Conference on Pattern Recognition, Copenhagen, Denmark, (1974), pp. 424–433.

[11] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: active contour models*, International Journal of Computer Vision, 1 (1988), pp. 321–331.

[12] D. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.

[13] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, Berkeley, Calif., 1967, University of California Press, pp. 281–297.

[14] K. E. G. Magnusson and J. Jalden, *A batch algorithm using iterative application of the Viterbi algorithm to track cells and construct cell lineages*, in Proceedings of the 9TH IEEE International Symposium on Biomedical Imaging, 2012, pp. 382–385.

[15] M. Maska, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. Espana, S. Venkatesan, D. M. W. Balak, P. Karas, T. Bolckova, M. Streitova, C. Carthel, S. Coraluppi, N. Harder, K. Rohr, K. E. G. Magnusson, J. Jalden, H. M. Blau, O. Dzyubachyk, P. Krizek, G. M. Hagen, D. Pastor-Escuredo, D. Jimenez-Carretero, M. J. Ledesma-Carbayo, A. Munoz-Barrutia, E. Meijering, M. Kozubek, and C. Ortiz-de Solorzano, *A benchmark for comparison of cell tracking algorithms*, Bioinformatics, 30 (2014), pp. 1609–1617.

[16] N. Milovsky, *The basics of game theory and associated games*. http://issuu.com/johnsonnick895/docs/game_theory_paper. Accessed: 2015-07-27.

[17] J. F. Nash, *Equilibrium points in n-person games*, Proceedings of the National Academy of Sciences of the United States of America, 36 (1950), pp. 48–49.

[18] N. Otsu, *A threshold selection method from gray-level histograms*, IEEE Transactions on Systems, Man, and Cybernetics, 9 (1979), pp. 62–66.

[19] M. Ovesny, P. Krizek, J. Borkovec, Z. K. Svindrych, and G. M. Hagen, *ThunderSTORM: a comprehensive ImageJ plug-in for PALM and STORM data analysis and super-resolution imaging*, Bioinformatics, 30 (2014), pp. 2389–2390.

[20] D. Pastor-Escuredo, M. A. Luengo-Oroz, L. Duloquin, B. Lombardot, M. Ledesma-Carbayo, P. Bourgine, N. Peyrieras, and A. Santos, *Spatio-temporal filtering with morphological operators for robust cell migration estimation in "in-vivo" images*, in 2012 9th IEEE International Symposium on Biomedical Imaging (ISBI), 2012, pp. 1312–1315. IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI 2012), Barcelona, Spain, MAY 2012.

[21] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen, *A survey of thresholding techniques*, Computer Vision, Graphics and Image Processing, 41 (1988), pp. 233–260.

[22] J. Selinummi, P. Ruusuvuori, I. Podolsky, A. Ozinsky, E. Gold, O. Yli-Harja, A. Aderem, and I. Shmulevich, *Bright field microscopy as an alternative to whole cell fluorescence in automated analysis of macrophage images*, PLOS ONE, 4 (2009).

[23] D. Svoboda and V. Ulman, *Generation of synthetic image datasets for time-lapse fluorescence microscopy*, in Image Analysis and Recognition, Pt II, vol. 7325 of Lecture Notes in Computer Science, 2012, pp. 473–482. 9th International Conference on Image Analysis and Recognition (ICIAR), Aveiro, Portugal, June 2012.

[24] A. Tsai, J. Yezzi, A., W. Wells, C. Tempany, D. Tucker, A. Fan, W. Grimson, and A. Willsky, *A shape-based approach to the segmentation of medical imagery using level sets*, IEEE Transactions on Medical Imaging, 22 (2003), pp. 137–154.

[25] M. Tscherepanow, F. Zoellner, A. Hillebrand, and F. Kummert, *Automatic segmentation of unstained living cells in bright-field microscope images*, in Advances in Mass Data Analysis of Images and Signals in Medicine, Biotechnology, Chemistry and Food Industry, Proceedings, vol. 5108 of Lecture Notes in Artificial Intelligence, 2008, pp. 158–172. 3rd International Conference on Mass Data Analysis of Signal and Images in Medicine, Biotechnology, Chemistry and Food Industry, Leipzig, Germany, July 14, 2008.

[26] T. Tse, L. Bradbury, J. W. L. Wan, H. Djambazian, R. Sladek, and T. Hudson, *A combined watershed and level set method for segmentation of brightfield*

cell images, Proceedings of SPIE Symposium on Medical Imaging: Image Processing, 7259 (2009).

[27] S. Yu and M. Berthod, *A game strategy approach for image labeling*, Computer Vision and Image Understanding, 61 (1995), pp. 32 – 37.

[28] F. Zernike, *Phase contrast, a new method for the microsopic observation of transparent objects*, Physica, 9 (1942), pp. 686–698.

[29] S. W. Zucker, *Region growing: childhood and adolescence*, Computer Graphics and Image Processing, 5 (1976), pp. 382–399.