Reconstruction of Incomplete Cell Paths through Three-Dimensional Segmentation

by

Maia Alicia Hariri

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, 2011

Abstract

Typical data used by biologists to track cells is obtained by recording a sequence of two-dimensional microscopy images over a finite period of time. However, in fluorescent microscopy images, cells may disappear for a few frames and then reappear. This can lead to confusion since the biologists have no way of predicting whether these cells were present in a previous frame or simply a result of cell division. When the set of images are stacked in chronological order, they form a three-dimensional image volume in which the disappearance of cells leads to broken cell paths. In this paper, we present two segmentation methods that are capable of reconstructing incomplete cell paths as a tool for tracking cells. The first model is inspired by geodesic active contours and Markov chains. The key idea is to generate a sequence of pseudo-random numbers that create a set of "invisible boundaries" within the 3D volume. Applying the geodesic segmentation algorithm with the knowledge of these boundaries will capture the gaps in the incomplete cell paths. The second approach performs 2D segmentation in a 3D framework using a similar formulation as the Chan-Vese level set segmentation. The 2D segmentation locates the cells that are visible in the image frames while the 3D segmentation captures the gap. We demonstrate the accuracy of our models both qualitatively and quantitatively by presenting the segmentation results of C2C12 cells in fluorescent images.

Acknowledgements

I would like to thank Justin W.L. Wan for all his help with this project. I would also like to thank Robert Sladek and Haig Djambazian from the Department of Medicine and Human Genetics at McGill University for generating the C2C12 fluorescent microscopy images. Finally, I would like to thank my classmates for their constant advice and encouragement.

Dedication

I dedicate this paper to my family and friends.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The study of cells is crucial in the development of science. Diseases like cancer and AIDS are direct consequences of cell mutations such as rapid cell growth, abnormal cell division, and large quantities of cell death. For this reason, biologists consistently rely on live cell imaging via light microscopy for medical research and monitoring of diseases. The idea is to use visible light to detect and enlarge small objects such as cells inside a given frame. With this research tool, cell biologists can extract detailed images of the interrelated structures and dynamics of biological systems [13].

Fluorescent microscopy is one of the several types of imaging techniques used by these biologists to study cell movement, cell growth, cell metabolism, cell differentiation, and cell death [11], [13]. Images from fluorescent microscopy are recorded in a time series, for example, taken every thirty minutes over a period of seven days. Typical experiments result in hundreds of images, each containing many cells that undergo movement, division, growth, and death, which render manual analysis extremely meticulous and time-consuming [13]. For illustration purposes, two consecutive fluorescent images from a sequence of eighty are shown in Figure 1.1. In fluorescent microscopy, the cells are tagged with a protein that exhibits fluorescence when exposed to light of different wavelengths. The fluorophore is the component of the protein that causes the molecule to be fluorescent. The result is an image in which the objects of interest, usually the nuclei of the cells, are easily distinguishable from the background because they appear as bright convex shapes. However, the amount of energy emitted by the fluorophores depends on the environment they are in. Therefore it is sometimes possible for the protein to not emit a visible light when the cell undergoes chemical changes during its division cycle. This property of fluorophores causes the cell to be missing from the frame of interest for a short period of time. Hence, given a sequence of fluorescent microscopy cell images obtained over a certain time interval, it is easy to

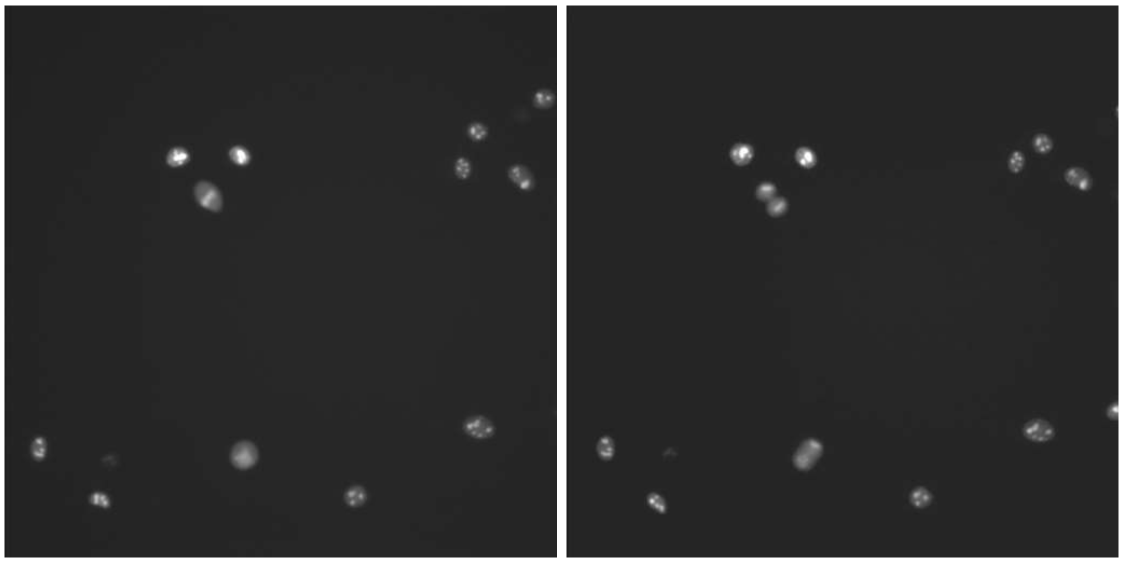
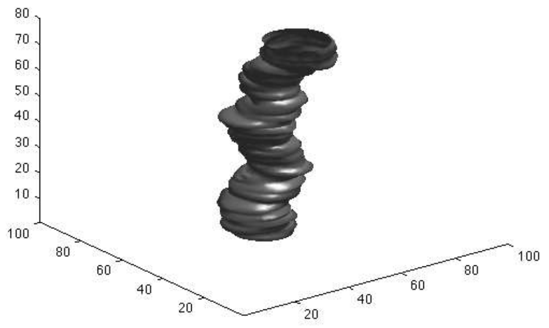


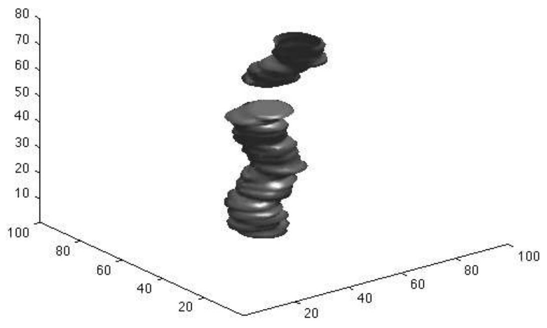
Figure 1.1: Fluorescent microscopy images of the same cell culture taken at different times

locate the positions of the various objects in each frame. The problem lies in tracking the origin of a cell when it reappears into the frame after disappearing for a short period of time. Although this is a problem that is often mentioned in the literature, a solution that does not involve manual intervention has not yet been addressed.

Image segmentation is a common tool used in the automated algorithm of cell tracking. By definition, it is the process of locating the boundaries of the objects in a given image. A common challenge for programmers is to develop a segmentation method that is able to detect edges as accurately as the human eye, despite gaps and broken boundaries in the actual image. The classic tracking method involves recording position and boundary information of the cells, at specified regular time intervals, using an appropriate segmentation technique. The identified boundaries in a given frame are then used as initial guesses for the position of the cells in the following frame [9]. In the case of a cell disappearing and reappearing, it is up to a specialist to identify the origin of the cell. This segment-and-track method only uses information from the previous image frame. However, the entire sequence of recorded fluorescent images is often available for analysis. In this paper, we will exploit the temporal information of the cells in order to reconstruct the missing segments when a cell disappears from the frame. The set of images obtained from fluorescent microscopy are stacked in chronological order to create a three-dimensional image volume. The sequence of cells form“tubes” in the image volume; for example, the stacking of one



(a) Complete cell path



(b) Incomplete cell path

Figure 1.2: Three-dimensional image volume of one cell

cell is shown in Figure 1.2. We define the case of a cell disappearing from the frame of interest, which can be seen in Figure 1.2(b), as an incomplete cell path because there is a gap in the 3D image volume. Applying a three-dimensional segmentation to the volume will capture the cell tubes and hence the locations of the cells at different times. However, this approach alone does not resolve the issue of tracking a cell when it reappears into the frame after disappearing for some time.

The goal of this project is to develop a technique for reconstructing incomplete cell paths through a three-dimensional segmentation. The algorithm should automatically be able to bridge gaps that are easily detectable by the human eye in any image volume. The input parameters may vary from one experiment to another, however, within a single experiment, no adjustments to the parameters should be necessary. We present two different approaches to solve this problem, "Random Walk Geodesic Active Contours" and "3D-2D Level Sets." The first method involves modifying the geodesic active contour model as well as the Metropolis algorithm from Markov Chain Monte Carlo methods to create a set of "invisible boundaries" that bridge the gaps in the 3D volume. The second algorithm modifies the active contours without edges model to perform a two-dimensional segmentation in a three-dimensional framework, such that the 2D segmentation captures the cells that appear in the image frames while the 3D segmentation reconstructs the incomplete cell path.

The rest of the paper is arranged as follows: Chapter 2 contains the background information on two well-known segmentation algorithms, geodesic active contours and active contours without edges, as well as a brief introduction to Markov Chain Monte Carlo methods; Chapter 3 describes the two different models investigated in this project for reconstructing incomplete cell paths; Chapter 4 presents and compares the results for both methods and Chapter 5 is the conclusion of this paper.

Chapter 2

Background

In this chapter, we describe the background information that was used in the development of the methods for reconstructing incomplete cell paths. The first algorithm was derived by combining geodesic active contours with Markov chain Monte Carlo methods, whereas the second is a modification of the active contours without edges segmentation technique. Hence, we first consider the two different level set models, geodesic active contours and active contours without edges, and present the discretization of the level set partial differential equation. Then we introduce a popular Markov Chain Monte Carlo method, called the Metropolis algorithm, for generating pseudo-random numbers.

2.1 Level Set Segmentation Methodology

The level set method introduced by Osher and Sethian [12] stems from a class of segmentation methods known as active contours. The basic idea for any active contour model is to evolve a curve over time so that it forms an outline around the object of interest. The classical active contour method is called snakes [8]. In this method, the evolving curve is defined as a parameterized set of points that are attracted towards the edges of the objects in an image. In particular, the snakes method is an energy minimization approach to segmentation such that the local minimum is obtained at the boundary of the object. Unlike the level set method, snakes cannot directly deal with changes in topology. Therefore, the main advantage of the level set method is that the active contour naturally splits and merges, allowing the simultaneous detection of several objects.

The level set method was derived using partial differential equations. The derivation involves Ω, an open bounded set in $\mathbb{R}^n$ with boundary $\partial\Omega$, and a level set function ϕ. Ω

corresponds to the object that we wish to segment, ϕ is an $(n+1)$-dimensional curve that is evolved over time, and the zero level set of ϕ defines $\partial\Omega$ when the stopping conditions are met. For example, in 3D, the boundary of the object is given by:

$$\partial\Omega = \{(x, y, z) \,|\, \phi(x, y, z) = 0\}.$$

More specifically, at any time t and position $\mathbf{x} = (x, y, z)$, the boundary of the current segmentation can be determined by solving

$$\phi(\mathbf{x}(t), t) = 0. \tag{2.1}$$

Furthermore, ϕ is a signed distance function with the following properties:

$$\begin{aligned} \phi(\mathbf{x}(t), t) &= 0 \quad \text{for } \mathbf{x} \in \partial\Omega \\ \phi(\mathbf{x}(t), t) &> 0 \quad \text{for } \mathbf{x} \in \Omega \\ \phi(\mathbf{x}(t), t) &< 0 \quad \text{for } \mathbf{x} \in \mathbb{R}^n \backslash (\Omega \cup \partial\Omega). \end{aligned}$$

A brief summary of the level set segmentation method is as follows. In order to determine the motion of the contour, the chain rule is used to differentiate (2.1) with respect to time [3]:

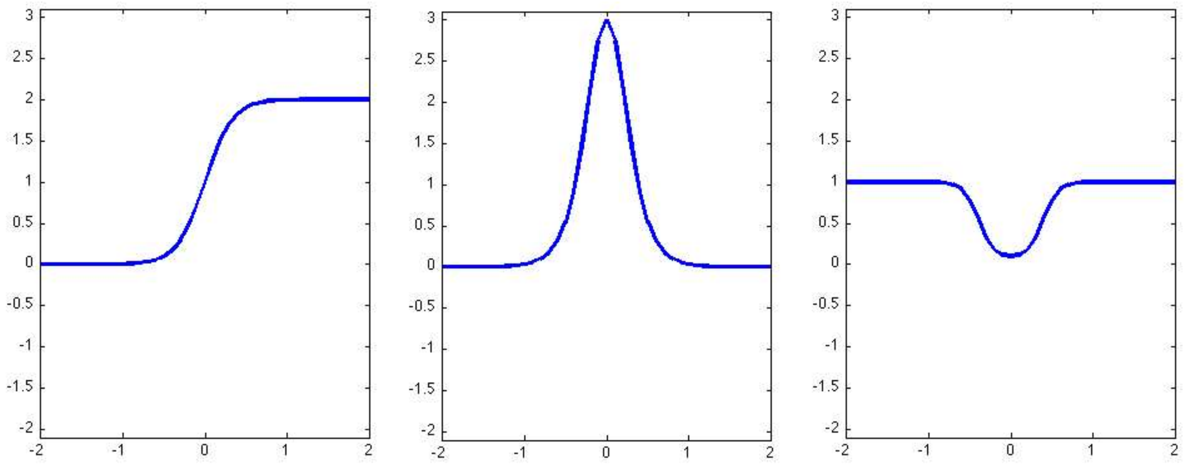
$$\frac{\partial\phi}{\partial t} + \nabla\phi(\mathbf{x}(t), t) \cdot \mathbf{x}'(t) = 0 \iff \frac{\partial\phi}{\partial t} + \left(\frac{\nabla\phi(\mathbf{x}(t), t)}{|\nabla\phi|} \cdot \mathbf{x}'(t)\right) |\nabla\phi| = 0.$$

Hence the curve propagates in a direction normal to itself with speed $F = -\vec{n} \cdot \mathbf{x}(t)$, where $\vec{n} = -\frac{\nabla\phi}{|\nabla\phi|}$ is the inward direction normal to the surface. Then the general level set equation is given by

$$\begin{cases} \frac{\partial\phi}{\partial t} + F|\nabla\phi| & = 0 \\ \phi(\mathbf{x}, 0) & = \phi_0 \end{cases} \tag{2.2}$$

where F dictates how fast the curve moves at each point in the image and ϕ_0 is the zero level set of the function ϕ at time $t = 0$.

The choice of speed function F depends on the level set model that is used. For example, information about the edges of an object in an image u can be extracted from the gradient of the image. More precisely, an image exhibits a rapid change in intensity values at the boundaries of an object. Hence the gradient method detects the edges by looking for the maximum values in the first derivative of the image. Consider the one-dimensional signal u illustrated in Figure 2.1. The edge is indicated by the jump in intensity in Figure 2.1(a). If we take the gradient of this signal, which in one dimension is just the first derivative



(a) u (b) $|\nabla u|$ (c) $g(|\nabla u|)$

Figure 2.1: Geometric interpretation of the edge stopping function g

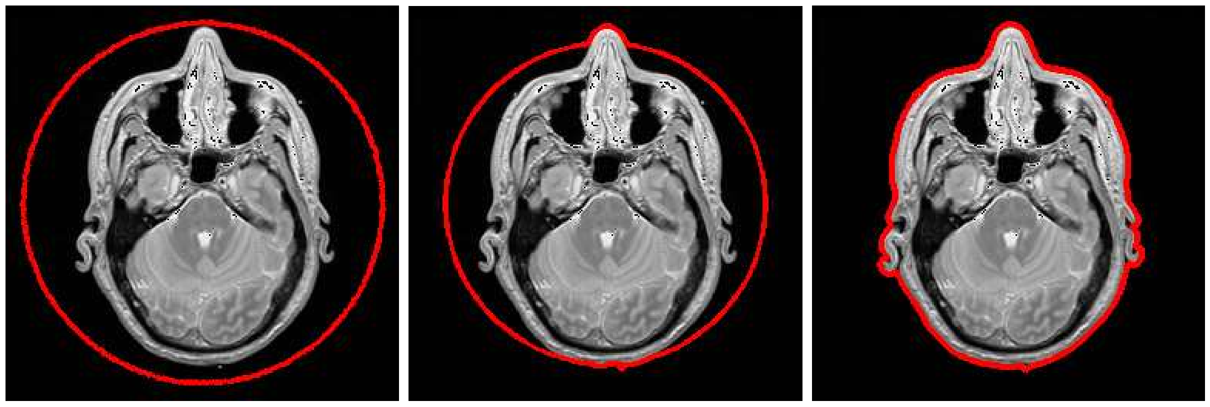


Figure 2.2: Evolution of a level set curve using the anisotropic diffusion model. Source: The Visible Human Project, MRI scans, Proton Density Image

with respect to the horizontal variable, we get the graph in Figure 2.1(b). The derivative exhibits a maximum at the centre of the edge in the original signal.

Because the curve should keep moving in the background of the image and stop evolving when it is close to the boundaries of the object, we define an edge stopping function g such that it is very small at the boundaries of an object and large everywhere else. That is, g is a strictly decreasing function such that $g \to 0$ as $|\nabla u| \to \infty$. Therefore, an appropriate edge stopping function is $g(r) = 1/(1 + r^p)$, where p is a positive integer. For illustration purposes, in Figure 2.1(c), observe that g is close to 0 at the edge of the signal and close to 1 in the background of the image. If F is chosen to be the edge stopping function

$$g(|\nabla u|) = \frac{1}{1 + |\nabla u|^2}, \tag{2.3}$$

then this is known as the anisotropic diffusion model, which is shown in Figure 2.2.

In this paper, we discuss two different possibilities for the choice of F. The first is called geodesic active contours and is similar to the anisotropic diffusion model in that it uses information about the edges in the image to calculate the propagation speed of the curve. The second is called active contours without edges and uses mean intensity values to split the image into two regions, the background and the objects.

2.1.1 Geodesic Active Contours

Caselles, Kimmel, and Sapiro [5] introduced a novel technique for detecting object boundaries called geodesic active contours in which the curves evolve over time according to intrinsic geometric measures. Let $\phi(s) : [0, 1] \to \mathbb{R}^n$ be a parameterized planar curve, α, β, and λ are constants, and u is the image in which we want to detect the object boundaries. Then the snakes approach tries to find the curve that minimizes the energy functional (2.4) such that the local minimum is obtained at the boundary of the object.

$$E(\phi) = \alpha \int_0^1 |\phi'(s)|^2 \, ds + \beta \int_0^1 |\phi''(s)|^2 \, ds - \lambda \int_0^1 |\nabla u(\phi(s))| \, ds. \tag{2.4}$$

The first two terms in (2.4) represent the internal energy of the system and control the smoothness of the curve. More specifically, the first term is an elastic term that discourages stretching and the second is a curvature term that discourages bending. The third term represents the external energy of the system and is responsible for attracting the curve towards the boundaries of the objects.

In [5], the authors set the rigidity constant β to zero and argue that the first term in (2.4) is sufficient to control the smoothness of the curve. Hence they reduce the energy functional to

$$E(\phi) = \alpha \int_0^1 |\phi'(s)|^2 \, ds - \lambda \int_0^1 |\nabla u(\phi(s))| \, ds. \tag{2.5}$$

Note that in this equation, minimizing E is equivalent to maximizing the second term while maintaining a certain smoothness to the curve. Hence we are trying to locate ϕ at the maxima of $|\nabla u|$. This is also the goal of the edge detector function defined in (2.3), so we can replace $-|\nabla u|$ in equation 2.5 with $g(|\nabla u|)$, creating a particular snakes model.

A geodesic curve is defined as a local minimal distance path between given points. For example, in two-dimensional Euclidean space, the shortest path between two points is a line segment given by the distance formula $d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$. The derivation of the geodesic active contour model involves Maupertuis' Principle[1] from dynamical systems to show that the solution to the particular snake model is given by a geodesic curve in a Riemannian space induced from the image u. Hence the energy minimization problem derived from the snakes model is reduced to a distance minimization problem in a Riemannian space where the functional is

$$L(\phi) = \int_0^1 g\big(|\nabla u(\phi(s))|\big)|\phi'(s)| \, ds. \tag{2.6}$$

In order to minimize (2.6), we need to compute the Euler-Lagrange of the above functional. The descent direction is parameterized by time $t > 0$ and $\kappa = \operatorname{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right)$ is the Euclidean curvature. The Euler-Lagrange equation for ϕ is given by

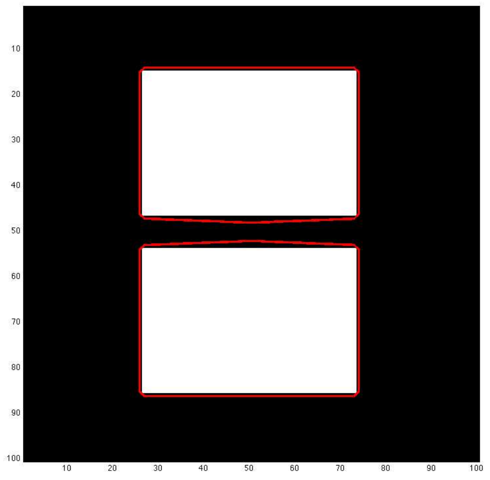
$$\frac{\partial \phi(t)}{\partial t} - g(\nabla u)\kappa|\nabla \phi| - \nabla g(\nabla u) \cdot \nabla \phi = 0. \tag{2.7}$$

In this paper, we use the following formula to compute curvature:

$$\kappa = \frac{1}{|\nabla \phi|^3}\big(\phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx} + \phi_x^2\phi_{zz} - 2\phi_x\phi_z\phi_{xz} + \phi_z^2\phi_{xx} + \phi_y^2\phi_{zz} - 2\phi_y\phi_z\phi_{yz} + \phi_z^2\phi_{yy}\big).$$

Furthermore, to deal with gaps in the boundaries of an object of the order of magnitude $\frac{1}{3c}$, the authors in [4] and [5] add a term which acts as a balloon force, requiring that the

[1]Let $U(\phi) = -\lambda g(\nabla u), \alpha = \frac{m}{2}$, and $p = m\phi$. Curves $\phi(s)$ in Euclidean space which are extremal corresponding to the Hamiltonian $H = \frac{p^2}{2m} + U(\phi)$, and have a fixed energy level E_0 (law of conservation of energy), are geodesics, with non-natural parameter, with respect to the new metric $(i, j = 1, 2) : g_{ij} = 2m\big(E_0 - U(\phi)\big)\delta_{ij}$ [5].


(a) $c = 0$

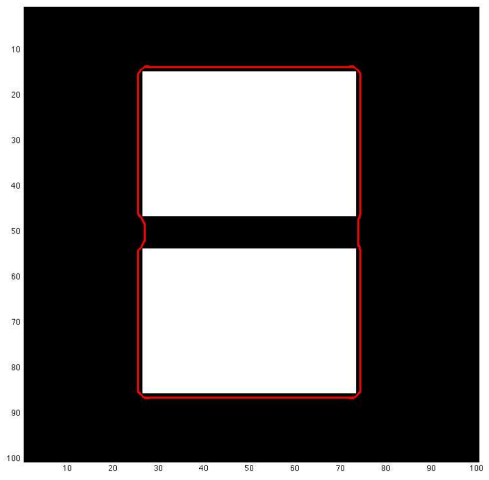

(b) $c = 0.87$

Figure 2.3: Geodesic segmentation of rectangles separated by a small gap of 8 pixels

region bounded by the level set maintain a certain area or volume. This term is called the constant velocity and depends on a positive real constant c. The suggested differential equation is

$$\frac{\partial \phi(t)}{\partial t} - g(\nabla u)\kappa|\nabla\phi| - \nabla g(\nabla u)\cdot\nabla\phi - g(\nabla u)c|\nabla\phi| = 0. \tag{2.8}$$

This can be rewritten in a way that resembles the level set equation:

$$\frac{\partial \phi(t)}{\partial t} + \Big(- g(\nabla u)(\kappa + c) - \nabla g(\nabla u)\cdot\frac{\nabla\phi}{|\nabla\phi|}\Big)|\nabla\phi| = 0. \tag{2.9}$$

Hence, setting $F = -g(\nabla u)(c+\kappa) - \nabla g(\nabla u)\cdot\frac{\nabla\phi}{|\nabla\phi|}$ yields the level set equation (2.2).

In this model, the evolution of the level set function does not uniquely depend on the edge stopping function g. In the case where the edges are not ideal, that is, there are different gradient values along the edges of the objects, the third term of (2.8), $\nabla(g)\cdot\nabla(\phi)$ will push the level set curve towards these boundaries. The geodesic active contour method is also particularly good at detecting objects with gaps in the boundary. For example, for the 2D case, in Figure 2.3(a), when $c = 0$, the evolving contour splits and does not include the gap in the segmented region. However in Figure 2.3(b), when $c = 0.87$, the final embedding function identifies both regions as one. It is important to note that in

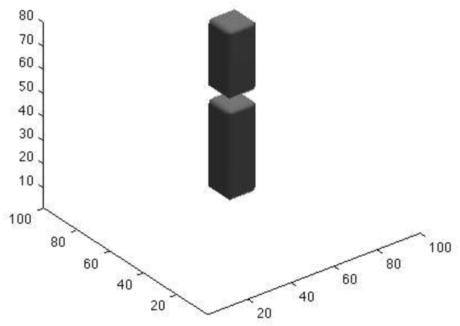


Figure 2.4: Geodesic segmentation of rectangular prisms separated by a small gap of 8 voxels with $c = 0.87$

the 3D case, as in Figure 2.4, including the constant velocity term has no effect on the segmentation.

2.1.2 Active Contours without Edges

Chan and Vese [6] presented a new model for active contours that detects objects whose edges are not necessarily defined by gradient. Instead of using an edge stopping function that relies on the gradient of the image u, the stopping term in [6] is based on the Mumford-Shah segmentation techniques. Assume that u can be separated into two regions of approximately piecewise-constant intensities, of distinct values u^{in}, which represents the object to be segmented, and u^{out}, which corresponds to the background of the image. Recall that $\partial\Omega$ defines the boundary of the object Ω. Then, in mathematical terms, we have

$$\begin{aligned} u(x,y,z) &\approx u^{in} \quad \text{for } (x,y,z) \in \Omega \cup \partial\Omega \\ u(x,y,z) &\approx u^{out} \quad \text{for } (x,y,z) \in \mathbb{R}^n \backslash (\Omega \cup \partial\Omega). \end{aligned}$$

In the three-dimensional implementation of the Chan-Vese model, a level set function is used to divide the image $u(x,y,z)$ into two regions where $\phi > 0$ is the inside and $\phi < 0$ is the outside. Let c_1 and c_2 be the average intensity values of u inside and outside ϕ

respectively, then for any curve ϕ, the following fitting terms are considered:

$$F_1(\phi) = \int_{inside(\phi)} \big(u(x,y,z) - c_1\big)^2 \, dxdydz$$
$$F_2(\phi) = \int_{outside(\phi)} \big(u(x,y,z) - c_2\big)^2 \, dxdydz.$$

These terms measure the closeness of the regions separated by $\phi > 0$ and $\phi < 0$ to u^{in} and u^{out} respectively. In fact, at the boundary of the object, we expect to have $F_1(\partial\Omega) + F_2(\partial\Omega) \approx 0$. Therefore, finding the boundary of the object is equivalent to finding the curve ϕ that minimizes the sum of the fitting terms:

$$\inf_\phi\{F_1(\phi) + F_2(\phi)\} \approx 0 \approx F_1(\partial\Omega) + F_2(\partial\Omega). \tag{2.10}$$

Figure 2.5 shows how $\partial\Omega$ minimizes (2.10).

In addition to the fitting terms in (2.10), Chan and Vese introduce parameters μ, ν, λ_1 and λ_2 to construct the following energy functional:

$$\begin{aligned} F(c_1, c_2, \phi) = {} & \mu \cdot \text{Length}(\phi) + \nu \cdot \text{Area}(inside(\phi)) \\ & + \lambda_1 \int_{inside(\phi)} \big(u(x,y,z) - c_1\big)^2 \, dxdydz \\ & + \lambda_2 \int_{outside(\phi)} \big(u(x,y,z) - c_2\big)^2 \, dxdydz. \end{aligned} \tag{2.11}$$

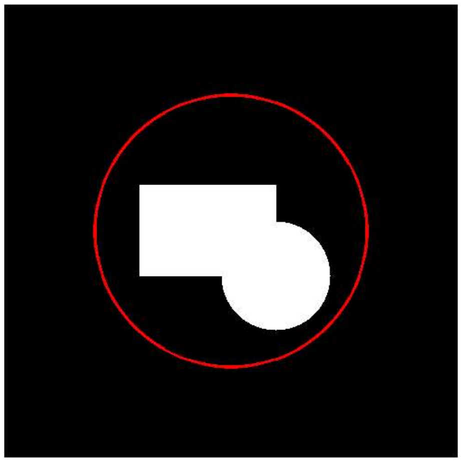
The last two terms in (2.11) are defined such that the energy is minimized when the level set surface is located right on the boundary of the object. The other two are regularization terms, the first is inserted to minimize the surface area and the second is meant to minimize the volume of the inside region.

Using the Heaviside function H and the delta Dirac measure δ, the Chan-Vese model minimizes the energy functional

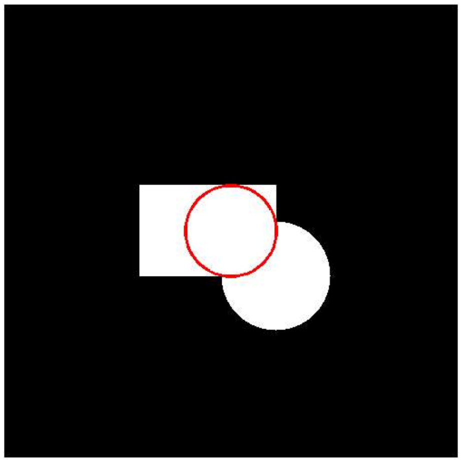
$$\begin{aligned} \min_\phi \quad & \mu \int_\Omega |\nabla H(\phi)| \, dxdydz \; + \; \nu \int_\Omega H(\phi) \, dxdydz \; + \; \lambda_1 \int_\Omega |u - c_1|^2 H(\phi) \, dxdydz \\ & + \lambda_2 \int_\Omega |u - c_2|^2 (1 - H(\phi)) \, dxdydz. \end{aligned}$$

The corresponding Euler-Lagrange equation for ϕ is given by

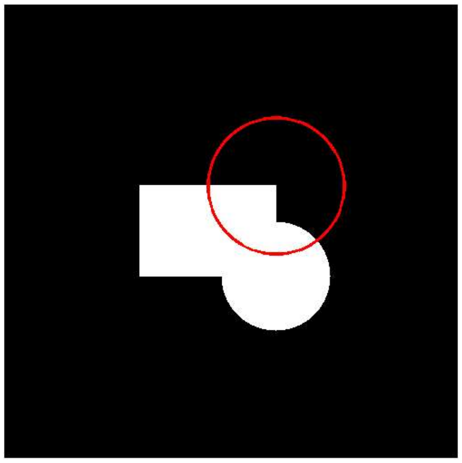
$$\frac{\partial\phi}{\partial t} - \delta(\phi)\left[\mu \cdot \text{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \nu - \lambda_1(u - c_1)^2 + \lambda_2(u - c_2)^2\right] = 0, \tag{2.12}$$



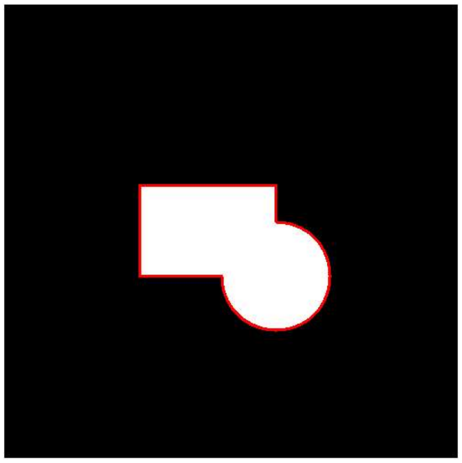
(a) $F_1(\phi) > 0$, $F_2(\phi) \approx 0$



(b) $F_1(\phi) \approx 0$, $F_2(\phi) > 0$



(c) $F_1(\phi) > 0$, $F_2(\phi) > 0$



(d) $F_1(\phi) \approx 0$, $F_2(\phi) \approx 0$

Figure 2.5: Fitting terms for all possible cases of the position of ϕ in 2D

with initial contour given by $\phi(x,y,z,0) = \phi_0(x,y,z)$. At the steady state, the zero level set of ϕ gives the boundary of the object. In order to extend the evolution to all level sets of ϕ, it is possible to replace $\delta(\phi)$ with $|\nabla\phi|$ and recalling that $\kappa = \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right)$, we rewrite equation (2.12) in a way that resembles the level set equation:

$$\frac{\partial\phi}{\partial t} + \left(-\mu\cdot\kappa + \nu + \lambda_1(u-c_1)^2 - \lambda_2(u-c_2)^2\right)|\nabla\phi| = 0. \tag{2.13}$$

2.1.3 Discretization of the Level Set PDE

In this project, the initial value partial differential equation (2.2) is solved using a finite difference method with the upwinding scheme. The approximate solution to the above PDE at iteration time t_m and voxel location (i,j,k) is denoted by ϕ^m_{ijk}. Therefore, we approximate $\phi(\mathbf{x}(t),t)$ at discrete times $t_0, t_1, ...,$ and positions (x_i, y_j, z_k) such that $t_{m+1} - t_m = \Delta t$, $x_{i+1} - x_i = \Delta x$, $y_{j+1} - y_j = \Delta y$, and $z_{k+1} - z_k = \Delta z$ are constants, using the following upwinding scheme:

$$\phi^{m+1}_{ijk} = \phi^m_{ijk} - \Delta t\big[\max(F_{ijk},0)\nabla^+ + \min(F_{ijk},0)\nabla^-\big],$$

such that

$$\begin{aligned}\nabla^+ = \big[&\max(D^{-x}_{ijk},0)^2 + \min(D^{+x}_{ijk},0)^2 + \max(D^{-y}_{ijk},0)^2 \\ &+ \min(D^{+y}_{ijk},0)^2 + \max(D^{-z}_{ijk},0)^2 + \min(D^{+z}_{ijk},0)^2\big]^{1/2}\end{aligned}$$

$$\begin{aligned}\nabla^- = \big[&\max(D^{+x}_{ijk},0)^2 + \min(D^{-x}_{ijk},0)^2 + \max(D^{+y}_{ijk},0)^2 \\ &+ \min(D^{-y}_{ijk},0)^2 + \max(D^{+z}_{ijk},0)^2 + \min(D^{-z}_{ijk},0)^2\big]^{1/2}\end{aligned}$$

where $D^{-x}_{ijk} = \frac{\phi^m_{ijk} - \phi^{m-1}_{ijk}}{\Delta x}$ is the formula for backward differencing in the x-direction and $D^{+x}_{ijk} = \frac{\phi^{m+1}_{ijk} - \phi^m_{ijk}}{\Delta x}$ is the formula for forward differencing in the x-direction. This upwinding scheme is stable if the following Courant-Friedrichs-Lewy (CFL) condition is satisfied:

$$\left(\max_\Omega F\right)\Delta t \le \min(\Delta x, \Delta y, \Delta z).$$

For simplification purposes, we will always force the 3D volume to be an $n \times n \times n$ cube so that $\Delta x = \Delta y = \Delta z$ and the CFL condition is $(\max_\Omega F)\Delta t \le \Delta x$.

At time t_0, we initialize ϕ to be a signed distance function. That is, we let

$$d(\mathbf{x}) = \min_{\mathbf{x}_I \in \partial\Omega} \left(||\mathbf{x} - \mathbf{x}_I|| \right)$$

such that $d(\mathbf{x}) = 0 \ \forall \mathbf{x} \in \partial\Omega$, where $\partial\Omega$ is the border between the inside, Ω^+, and the outside, Ω^-, of the embedding function. Then

$$\phi(\mathbf{x}) = \begin{cases} 0 & \forall \mathbf{x} \in \partial\Omega \\ -d(\mathbf{x}) & \forall \mathbf{x} \in \Omega^- \\ d(\mathbf{x}) & \forall \mathbf{x} \in \Omega^+ \end{cases} .$$

In particular, for this project, ϕ_0 is constructed such that its zero level set is a sphere that surrounds the entire 3D image volume. Furthermore, every fifty iterations of the level set method, say at times $t_0 + 50$, $t_0 + 100$, and so forth, the embedding function is reinitialized to be a signed distance function that satisfies the properties descried above.

The number of time steps is not predefined by the user. Instead, we run the upwinding scheme until no more changes are made to ϕ from one time step to another. More precisely, at a given time step t_m, we let $\phi\left((x, y, z), t_m\right) = \phi_m$, and run the algorithm until the following stopping condition is met:

$$|\phi_{m+1} \geq 0| \simeq |\phi_m \geq 0|.$$

2.2 Markov Chain Monte Carlo Methods

In this section, we move away from image processing techniques to introduce the statistical background required in the derivation of the random walk geodesic active contour method.

Over the past 17 years, Markov Chain Monte Carlo (MCMC) techniques have taken Bayesian statistics to new heights by providing a universal tool for dealing with integration and optimization problems [7]. In the following sections, we define Markov Chains and present a common algorithm for generating a sequence of discrete random numbers.

2.2.1 Markov Chains

A *stochastic process* is a sequence of random variables $\{X(t),\, t \in T\}$, where t is a parameter in a set T. The state of the process at time t is called $X(t)$ and the set of all possible realizations of $X(t)$ describes the state space denoted by $\mathcal{S}$. A *discrete time stochastic process* is a stochastic process in which the parameter and state spaces are discrete, that is $T = \{0, 1, ...\}$ and $\mathcal{S} = \{0, 1, ...\}$ [7].

Definition 2.2.1 (Markov Chain) *A discrete time stochastic process* $\{X_t = X(t),\ t = 0, 1, ...\}$ *is said to be a Markov chain if the conditional distribution of the future state* $X(t_k)$ *given the present and past history of* $X(t)$*, i.e.* $X(t_{k-1}) = x_{k-1}, ..., X(t_1) = x_1$*, only depends on the present state* $X(t_{k-1}) = x_{k-1}$*. That is, for any set of* k *time points* $t_1 < t_2 < ... < t_k$ *in* T*,*

$$P\big(X(t_k) \le x_k \,|\, X(t_{k-1}) = x_{k-1}, ..., X(t_1) = x_1\big)$$
$$= P\big(X(t_k) \le x_k \,|\, X(t_{k-1}) = x_{k-1}\big).$$

The one-step *transition probability function* of a Markov chain defines the probability distribution of the next state given the present state:

$$p_{ij} = P(X_k = j \,|\, X_{k-1} = i),$$
$$\sum_{j \in \mathcal{S}} p_{ij} = 1.$$

If any of the transition probabilities change with time, that is $p_{ij}^{(n)} \neq p_{ij}^{(m)}$ for some $m, n \in T$, then the Markov chain is called *time inhomogeneous*, otherwise, the chain is called *stationary*.

The matrix of these transition probabilities shown below is square and is called the *transition matrix*. It is a stochastic matrix, hence by definition, $p_{ij} \geq 0 \,\forall\, i, j \in \mathcal{S}$ and $\sum_j p_{ij} = 1$.

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} & \cdots \\ p_{21} & p_{22} & p_{23} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Consider the simple example of a person, Adam, standing on one of a sequence of four tiles as illustrated in Figure 2.2.1. From one time step to another, Adam may move to a tile that is directly beside the one he is already on, or he can remain at his current location. In this case, $\mathcal{S} = \{1, 2, 3, 4\}$ and the transition matrix indicates the probability of the person moving from his current position $i \in \mathcal{S}$ to another location $j \in \mathcal{S}$. Assuming there are no biases between the tiles, the resulting probability transition matrix is

$$P = \begin{pmatrix} {}^1/_2 & {}^1/_2 & 0 & 0 \\ {}^1/_3 & {}^1/_3 & {}^1/_3 & 0 \\ 0 & {}^1/_3 & {}^1/_3 & {}^1/_3 \\ 0 & 0 & {}^1/_2 & {}^1/_2 \end{pmatrix}.$$

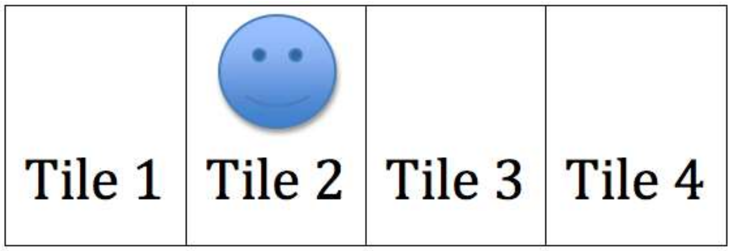


Figure 2.6: Transition matrix illustration

The first row of the matrix can be read as, given that Adam is on Tile 1, the probability that he will stay on his current tile is 1/2, the probability that he will move to Tile 2 is 1/2, and the probability that he will move to any other tile is 0. On the other hand, assume that there is a lion on Tile 3, so that Adam is inclined to stay away from that particular location, then a possible transition matrix is

$$P = \begin{pmatrix} 0.8 & 0.2 & 0 & 0 \\ 0.8 & 0.2 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2.2.2 Generating a Discrete Random Number

Let X be a discrete random variable with state space $\mathcal{S} = \{a_1, a_2, ..., a_k, ...\}$ and respective probabilities $p_1, p_2, ..., p_k, ...$ such that $\sum_k p_k = 1$. To generate a random number from $\mathcal{S}$, we partition the interval $[0, 1]$ into subintervals $I_1, I_2, ..., I_k, ...$ based on the densities [7]. That is, for $k = 1, 2, ...$

$$\begin{aligned} I_k &= (F_{k-1}, F_k], \\ F_0 &= 0, \\ F_k &= p_1 + p_2 + ... + p_k. \end{aligned}$$

Each subinterval corresponds to a single value for X; in particular I_k corresponds to the value a_k. Next, we randomly generate a value U from the uniform distribution $U(0, 1)$, which is identical to randomly picking a number between 0 and 1. The interval I_k to which

U belongs indicates the generated value for X, a_k. In summary,

$$X = \begin{cases} a_1 & \text{if} \quad U \in I_1 \\ a_2 & \text{if} \quad U \in I_2 \\ a_3 & \text{if} \quad U \in I_3 \\ \vdots & \vdots \end{cases}.$$

More specifically, in the following chapters of this paper, to generate X from a given row i in the probability transition matrix P, that is $X \sim P(i,:)$, we randomly generate $U \sim U(0,1)$ and set

$$X = \begin{cases} 1 & \text{if} \quad U \leq p_{i1} \\ 2 & \text{if} \ p_{i1} < U \leq p_{i2} \\ 3 & \text{if} \quad p_{i2} < U \leq p_{i3} \\ \vdots & \vdots \end{cases}.$$

For example, if

$$P = \begin{pmatrix} 0.8 & 0.2 & 0 & 0 \\ 0.8 & 0.2 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and we want to generate X from $P(2,:)$. We first generate $U \sim U(0,1)$, then

$$X = \begin{cases} 1 & \text{if} \quad U \leq 0.8 \\ 2 & \text{otherwise} \end{cases}.$$

2.2.3 Metropolis Algorithm

One of the most popular MCMC methods is the Metropolis algorithm. The idea was first introduced by Metropolis, Rosenbluth, and Teller as a method for the efficient simulation of the atomic energy levels in a crystalline structure [10]. The algorithm was later generalized by Hastings to focus on statistical problems. The goal of the Metropolis algorithm is to obtain a sample from a target distribution $\pi_j = P(X = j), j \in \mathcal{S}$. Suppose there exists a Markov chain with stationary distribution $\pi = (\pi_j\,; j \in \mathcal{S})^T$ and choose a symmetric stochastic matrix $Q = (q_{ij})_{i,j\in\mathcal{S}}$, then the Metropolis algorithm starts with a proposed state i and decides whether it moves to a new state j based on a Bernoulli distribution.

Algorithm 1 Metropolis

```
1: Start with $n = 1$ and $X_0 \in \mathcal{S}$
2: Set $i = X_{n-1}$ where $i \in \mathcal{S}$
3: Generate $j$ from the probability distribution $\{q_{ij}\,; j \in \mathcal{S}\}$
4: Set $r = \frac{\pi_j}{\pi_i}$
5: If $r \geq 1$ set $X_n = j$, otherwise generate $u \sim U(0,1)$. If $u < r$ set $X_n = j$, else set $X_n = X_{n-1}$
6: Set $n = n + 1$
7: Repeat from step 2 until the desired sample size is achieved.
```

Chapter 3

Methodology

In this chapter, we present two different methods for segmenting a gap in a three-dimensional image volume. The first is a three step algorithm which involves modifying the geodesic active contour model so that the constant velocity term is dependent on voxel location, as opposed to a constant c. This method can also be used to segment larger gaps in a two-dimensional image. The second is a 3D-2D segmentation model that performs two-dimensional segmentation in a three-dimensional framework using the active contours without edges model. For both of these methods, we describe the process of capturing gaps in the z-direction, however, the models can be adapted to capture gaps in any direction with a simple adjustment to certain parameters.

3.1 Random Walk Geodesic Active Contours

The constant velocity term in (2.8) is not sufficient to detect gaps in the z-direction of the volume. Thus we must manipulate the constant c term so that it is a function of spatial position, hence no longer a constant. In particular, we want to force the contour to stop at an "invisible boundary" that will be determined by a modified version of the Metropolis algorithm.

Recall the speed function used in the geodesic active contour model,

$$F = -\bigg(g(\nabla u)(c + \kappa) + \nabla g(\nabla u) \cdot \frac{\nabla \phi}{|\nabla \phi|} \bigg).$$

In the case where there are different gradient values along the edges of the objects, the second term is meant to push the level set function towards these non-ideal boundaries.

However, in reconstructing incomplete cell paths, the edges of the missing cells are non-existent, so they do not fall under the category of non-ideal boundaries. Hence, we simply need to worry about the first term in F, which is the edge stopping function. At a given voxel location (x, y, z), if $c(x, y, z) + \kappa(x, y, z) = 0$, then the contour is forced to stay still. The idea behind this method is to set $c(x, y, z) = -\kappa(x, y, z)$ for all (x, y, z) on the "invisible boundary." We propose three steps to implement this model:

1. Identify all possible gap locations by segmentation and clustering.
2. Use an algorithm inspired by Metropolis to locate the "invisible boundaries" within these gaps.
3. Solve the level set equation with speed function $F = -\big(g(\nabla u)(c+\kappa)+\nabla g(\nabla u)\cdot\frac{\nabla\phi}{|\nabla\phi|}\big)$, where $c = -\kappa$ for all points on the "invisible boundary."

These steps are explained in detail in the following three sections.

3.1.1 Locating the Gaps

In order to locate a possible gap region in a 3D image volume, we first use segmentation to divide the image volume into clusters based on voxel location and intensity value.

There are known segmentation techniques, such a k-means clustering, that will simultaneously segment the image and divide it into different sections. However, the number of clusters needs to be specified by user input and these methods are quite slow if the centres of the sections are unknown.

A simple way to separate the image into clusters is to first use an active contour method from Section 2.1 to divide the graph into two sections, $A = \{(x, y, z) \,|\, \phi(x, y, z) \geq 0\}$, representing the objects, and $B = \{(x, y, z) \,|\, \phi(x, y, z) < 0\}$, corresponding to the background. Next, each voxel, including the ones in the background, are labeled with an integer value ranging from 1 to K, where K is the total number of clusters. First and foremost, all the points in the background belong to the same group, therefore they are labeled 1. For the points inside the objects, the grouping is as follows: any two neighbouring voxels must belong to the same cluster C_k. The clustering algorithm below is derived from the Region Growing method of segmentation, however, the number of clusters need not be specified. Note that a non-boundary point has 26 neighbours. We call the clustered image u_c. The result of this algorithm is illustrated in Figure 3.1(b) where each colour in u_c represents a different group of pixels.

Algorithm 2 Clustering

```
Require: $A = \{(x,y,z) \,|\, \phi(x,y,z) \geq 0\}$
 1: $k \leftarrow 1$
 2: for each point $a \in A$ do
 3:     $k \leftarrow k+1$
 4:     Remove $a$ from $A$ and add it to $C_k$
 5:     repeat
 6:         for each voxel in the set $C_k$ do
 7:             for each of its neighbours $n_a$ that are not in $C_k$ do
 8:                 if $n_a \in A$ then
 9:                     Remove $n_a$ from $A$ and add it to $C_k$
10:                 end if
11:             end for
12:         end for
13:     until no pixels are added in the last pass
14: end for
15: Return $C_2, C_3, ..., C_K$
```

For each cluster C_k, $k \in \{1, ..., K\}$, an average intensity value μ_k and a centre location $\mathbf{p}_k = (\bar{x}_k, \bar{y}_k, \bar{z}_k)$ are calculated. Let I_k be an indicator function such that

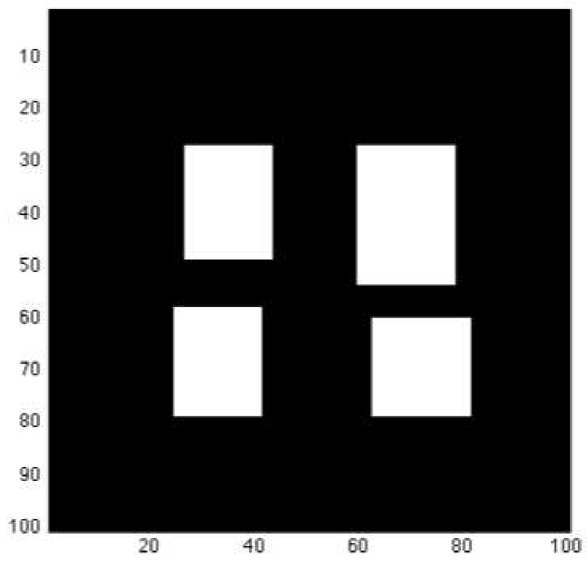
$$I_k(x,y,z) = \begin{cases} 1 & \text{if } u_c(x,y,z) = k \\ 0 & \text{otherwise.} \end{cases}$$

Then

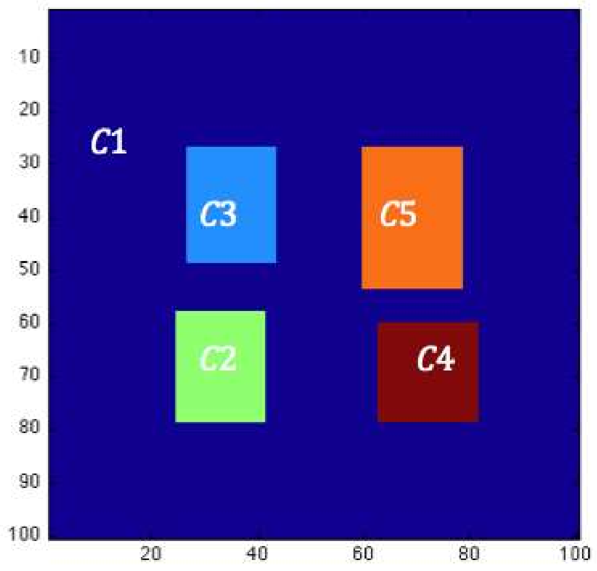
$$\mu_k = \frac{\sum_{x,y,z} I_k(x,y,z) \cdot u(x,y,z)}{\sum_{x,y,z} I_k(x,y,z)}, \; \mathbf{p}_k = \frac{\sum_{x,y,z} I_k(x,y,z) \cdot (x,y,z)}{\sum_{x,y,z} I_k(x,y,z)}.$$

The next step in finding a possible gap location is to identify all the points that are not in C_k but are close enough to be considered part of the cluster.

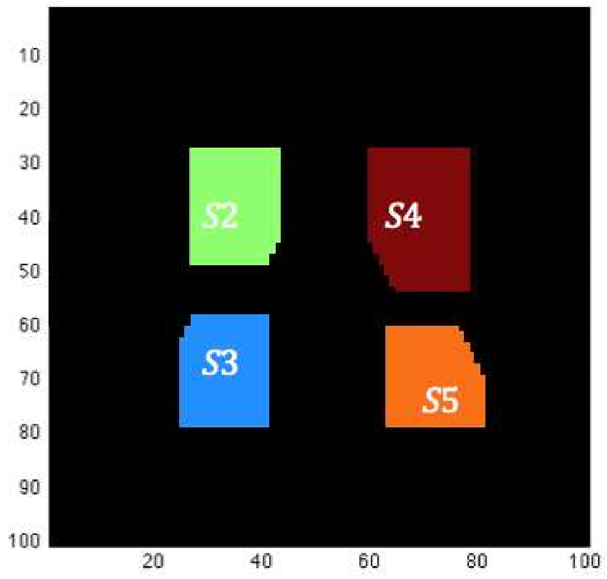
Once a clustered image u_c is attained, we use a measure of "closeness" based on distance and intensity values to determine whether two voxels belonging to different clusters should actually belong to the same one. More specifically, we need to determine whether a point (x,y,z) such that $u_c(x,y,z) \neq k$ should be part of C_k. This decision is influenced by the distance between (x,y,z) and $\mathbf{p}_k$ and the difference in intensity values, $|u(x,y,z) - \mu_k|$. If (x,y,z) is relatively close to C_k in terms of distance and intensity value, then it should also belong to that cluster. We let S_k be the set of all such points. Our measure of closeness is



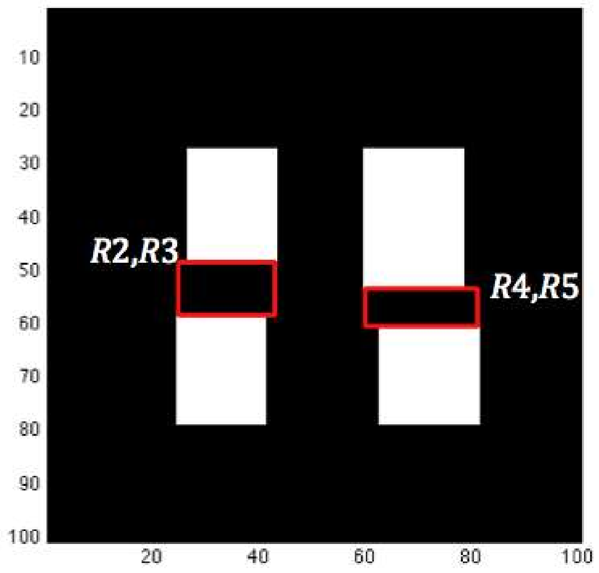
(a) Original image u



(b) Clustered image u_c



(c) Required connections



(d) Identified gap regions

Figure 3.1: 2D Example of locating the gaps

given in (3.1)

$$D_k(x,y,z) = w_1 \cdot |x - \bar{x}_k| + w_2 \cdot |y - \bar{y}_k| + w_3 \cdot |z - \bar{z}_k| + w_4 \cdot |u(x,y,z) - \mu_k|, \tag{3.1}$$

where $u_c(x,y,z) \neq k$ and $w_i \in \mathbb{R}$ for $i = 1, ..., 4$. Note that we do not compute distance using the Euclidean formula $d(\mathbf{a}, \mathbf{b}) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2}$ because all three directions may not be equally as important. For example, in our case of incomplete cell paths, the gaps only appear in the z-direction, therefore the vertical distance between two voxels is the most meaningful. Furthermore, two points with a very large difference in intensity values should obviously not belong to the same section. Therefore, in order to avoid labeling background points as part of the objects in the image, w_4 is set to be significantly larger than the other weights. For example, we choose $w_1 = w_2 = 1$, $w_3 = 2$ and $w_4 = 100$.

Information about the volume of the cluster k is used to compute a maximum allowed closeness, $D_{max,k}$, such that $(x,y,z) \in S_k \iff D_k(x,y,z) \leq D_{max,k}$. Let h be the height of the largest gap we wish to be capable of identifying and $d_{k,x}$, $d_{k,y}$, and $d_{k,z}$ be the maximum length, width, and height of C_k, respectively:

$$\begin{aligned}
d_{k,x} &= \big(\max_{u_c}\{x \,|\, u_c(x,y,z) = k\} - \min_{u_c}\{x \,|\, u_c(x,y,z) = k\}\big), \\
d_{k,y} &= \big(\max_{u_c}\{y \,|\, u_c(x,y,z) = k\} - \min_{u_c}\{y \,|\, u_c(x,y,z) = k\}\big), \\
d_{k,z} &= \big(\max_{u_c}\{z \,|\, u_c(x,y,z) = k\} - \min_{u_c}\{z \,|\, u_c(x,y,z) = k\}\big).
\end{aligned}$$

Furthermore, because the voxel intensity values within the 3D image are scaled to be between 0 and 1, the difference in intensity values between two points that should belong to the same cluster is close to 0, whereas it is close to 1 for two points that should clearly not be clustered together. Hence, we use

$$D_{max,k} = w_1 \cdot \frac{d_{k,x}}{2} + w_2 \cdot \frac{d_{k,y}}{2} + w_3 \cdot \left(\frac{d_{k,z}}{2} + h\right) + w_4 \cdot 0. \tag{3.2}$$

The process to locate the points in S_k for $k = \{1, .., K\}$ is described in Algorithm 3. The results of this algorithm applied to a simple 2D example with $h = 10$ pixels are shown in Figure 3.1(c). This time, the different colours indicate the cluster that each pixel should be connected to. For example, all the light blue pixels labeled S_3 in Figure 3.1(c) belong to the green cluster labeled C_2 in Figure 3.1(b). However, Algorithm 3 determined that they are all close enough in distance and intensity value to also belong to C_3.

The final step in this section is to locate a possible gap region, R_k, for each cluster k. To do this, we simply take the region bounded in between C_k and S_k. This is illustrated in

Algorithm 3

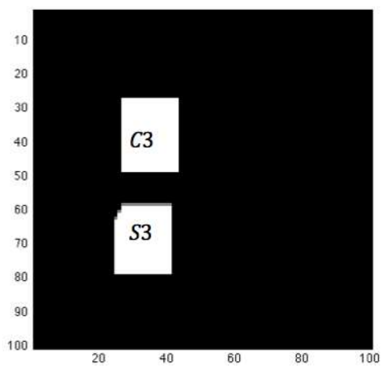
```
1: for $k = 1 \rightarrow K$ do
2:    Compute $D_{max,k}$ as in equation 3.2
3:    for all $(x, y, z) \in u$ such that $u_c(x, y, z) \neq k$ do
4:       Compute $D_k(x, y, z)$ as in equation 3.1
5:       if $D_k(x, y, z) \leq D_{max,k}$ then
6:          Add voxel $(x, y, z)$ to the set $S_k$
7:       end if
8:    end for
9: end for
10: return $S_1, S_2, ...S_k$
```

Figure 3.1(d) where the areas within the red contours are the identified gap regions. Note that, in this case, the gap regions identified by (C_2, S_2) and (C_4, S_4) are identical to those identified by (C_3, S_3) and (C_5, S_5) respectively.

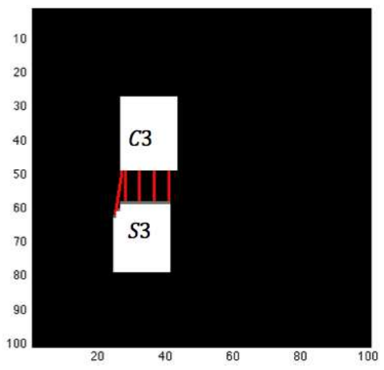
3.1.2 Identifying the Invisible Boundaries

Given a cluster C_k and a set of points S_k containing all the voxels that should be connected to C_k, we propose a method for creating a link that bridges the gap between these two sections.

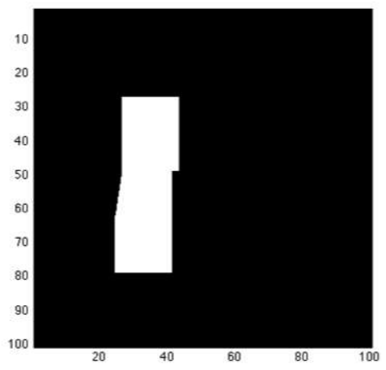
The most obvious solutions to this problem is to draw a minimal distance line connecting each point on the surface of S_k, e.g. the gray line at the top of S_3 in Figure 3.2(a), to a point in C_k, which is basically linear interpolation. In this case, neither step 2 nor step 3 of our summarized model is required. This process is illustrated in Figure 3.2(b) where lines are drawn to connect S_3 to C_3 from the example in Figure 2.5. However, this method only uses information from two slices in the 3D volume and assumes that the motion of a cell is linear. It is important to note that a cell begins to move in response to an external signal in its surrounding environment [2]. The cell senses the signal by spatially recognizing concentration gradients and tries to move in the direction of greatest concentration. Unfortunately, it is impossible for a cell to pick the correct direction without randomly sampling its surroundings. Therefore a stochastic process is often used to model cell movement. In an attempt to maintain the random motion of the cell within the gap, we modify the Metropolis algorithm from Section 2.2.3 to create a random walk from S_k to C_k. The idea is to generate a sequence of N points $\{X(t), t \in T\}$ where $T = \{1, ..., N\}$ connecting a point $(x_0, y_0, z_0) \in S_k$ to a point $(x_N, y_N, z_N) \in C_k$. At each time step t,



(a) C_3 and S_3



(b) Minimal distance lines



(c) S_3 connected to C_3

Figure 3.2: Bridging the gap between S_3 and C_3

we generate $(\bar{x}, \bar{y}, \bar{z})$ from a set of probability distributions and then accept or reject that point based on a Bernoulli distribution.

Because the gaps that we are dealing with are uniquely in the z-direction, the random walk should only move up, if S_k is below C_k, or down, if S_k is above C_k. Without loss of generality, we assume that S_k is below C_k and explain the steps for the "Random Walk Up" algorithm. The motion of the cells occurs in the x-y plane, hence we need to determine relevant probability transition matrices, P_x and P_y, for both of the variables x and y. Note that we do not have the appropriate data to be able to create an exact stochastic matrix that depicts the motion of the cell. Instead, we require that our model satisfy the following three properties:

1. The path must move more or less randomly towards the closest point in C_k.
2. The path must be continuous, that is $X(t_{i-1})$ and $X(t_i)$ should be neighbours.
3. The path must remain inside the region R_k.

Let $X(t_i) = (x_i, y_i, z_i)$. First, we set $\bar{z} = z_{i+1}$. Then, to satisfy the quality of randomness in property 1, we allow the path to move in the x-direction with uniform distribution, at all time $t \in T$. However, to maintain continuity we must have

$$x_{i+1} = \begin{cases} x_i - 1 & \text{with probability } {}^1/_3 \\ x_i & \text{with probability } {}^1/_3 \\ x_i + 1 & \text{with probability } {}^1/_3 \end{cases} .$$

Hence the Markov chain is stationary and the corresponding $n \times n$ transition matrix is

$$P_x = \begin{pmatrix} {}^1/_2 & {}^1/_2 & 0 & \cdots & 0 & 0 & 0 \\ {}^1/_3 & {}^1/_3 & {}^1/_3 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & {}^1/_3 & {}^1/_3 & {}^1/_3 \\ 0 & 0 & 0 & \cdots & 0 & {}^1/_2 & {}^1/_2 \end{pmatrix}.$$

We use this transition matrix to generate $\bar{x}$ from $P_x(x_i, :)$. Given $\bar{x}$ and $\bar{z}$, we determine $\bar{y}$ from the set $Y = \{y_i - 1, y_i, y_i + 1\}$. For $j = 1, ..., 3$, we compute the shortest distance from the point $(\bar{x}, \bar{y}_j, \bar{z})$ to C_k, where $\bar{y}_j \in Y$ and $\bar{y}_1 \neq \bar{y}_2 \neq \bar{y}_3$. Let d_1, d_2, and d_3 be these distances such that $d_1 \leq d_2 \leq d_3$. Then we have

$$\bar{y} = \begin{cases} \bar{y}_1 & \text{with probability } p_1 \\ \bar{y}_2 & \text{with probability } p_2 \\ \bar{y}_3 & \text{with probability } p_3 \end{cases},$$

with $p_1 \geq p_2 \geq p_3$ and $p_1 + p_2 + p_3 = 1$. Note that, in this case, the point that is closest to cluster C_k is given the highest probability and the resulting transition matrix is time inhomogeneous because it is a function of distance. In this project, we manually choose the values of p_1, p_2, and p_3. The closer p_1 is to 1, the more the random walk resembles a minimal distance line. More information about the choice of p_i is given in Algorithm 4. P_y is a sparse matrix similar to P_x in which each non-boundary row has three consecutive elements. Without loss of generality, assume that $\bar{y}_1 = y_i - 1$, $\bar{y}_2 = y_i$, and $\bar{y}_3 = y_i + 1$, then

$$\begin{aligned} P_y(1,\, 1:2) &= (p_1, p_2), \\ P_y(i,\, i-1 : i+1) &= (p_1, p_2, p_3) \,\text{for}\, i = 2, ..., n-1, \text{and} \\ P_y(n,\, n-1 : n) &= (p_1, p_2). \end{aligned}$$

Finally, to restrict the third property, we only accept the point $(\bar{x}, \bar{y}, \bar{z})$ as a vertex in our random walk if it is inside the possible gap region R_k. The above steps are summarized in Algorithm 4.

Algorithm 4 RandomWalkUp

```
Require: $N$ is the length of the random walk and $(x_0, y_0, z_0) \in S_k$ is the starting point of
    the walk.
Ensure: $X$ is a list of $N$ coordinates of the random walk
 1: $X(1,:) = (x_0, y_0, z_0)$
 2: for $i = 1 \to N-1$ do
 3:     $\bar{z} \leftarrow X(i,3) + 1$ {for the Random Walk Down, this line is $\bar{z} \leftarrow X(i,3) - 1$}
 4:     Generate $\bar{x}$ from $P_x\big(X(i,1),:\big)$ as in section 2.2.2
 5:     Compute $d_1$, $d_2$, and $d_3$ as defined above
 6:     if $d_1 = d_2 = d_3$ then
 7:         Set $p_1 = p_2 = p_3 = \frac{1}{3}$
 8:     else if $d_1 = d_2 \neq d_3$ then
 9:         Set $p_1 = p_2 = p$ and $p_3 = 1 - p$ for some $0.25 < p \leq 0.5$
10:     else
11:         Set $p_1 > 0.5 > p_2 \geq p_3$
12:     end if
13:     Generate $\bar{y}$ from $P_y\big(X(i,2),:\big)$ as in section 2.2.2
14:     if $(\bar{x}, \bar{y}, \bar{z}) \in R_k$ then
15:         $X(i+1,:) \leftarrow (\bar{x}, \bar{y}, \bar{z})$
16:     else
17:         $X(i+1,:) \leftarrow X(i,:)$
18:     end if
19: end for
20: return X
```

It is important to note that the boundaries of R_k are determined using information about the entire cluster C_k. Hence the range of motion of the path is not restricted by one slice in the 3D volume, as the linear interpolation method. Instead, the cells in the gap are allowed to move around in the frame of interest as freely as any other cell in the previous or subsequent slices.

To create the "invisible boundaries," for each set (C_k, S_k, R_k) in the image u, we identify all the points on the face of S_k that are closest to C_k and use them as input in the Random Walk Algorithm. If there are n_k such points, then we create n_k paths. However, we only consider the paths that terminate somewhere in C_k, as opposed to the background of the image. All the points in the random walks from S_k to C_k are considered part of the "invisible boundaries."

3.1.3 Final Segmentation of the Image

The result of the previous section is a logical $n \times n \times n$ matrix, L, indicating the voxel locations of the "invisible boundaries" in the image. At every iteration time t_m and voxel location (i, j, k) of the level set method presented in section 2.1, we denote the curvature, κ, and the speed function, c, by κ_{ijk}^m and c_{ijk}^m respectively, and set

$$c_{ijk}^m = \begin{cases} -\kappa_{ijk}^m & \text{if } L(i,j,k) = 1 \\ 0 & \text{otherwise} \end{cases} .$$

A complete algorithm of the "Random Walk Geodesic Active Contours" model is given in Algorithm 5.

It is important to note that the "invisible boundaries" do not form a solid contour. Out of all the random walks generated, only the ones connecting a point in S_k to one in C_k are used. Hence the region formed by the invisible boundaries contains little holes, especially for large gap sizes. For this reason, it is important that a final 3D segmentation is performed on the image; the curvature term will ensure the smoothness of the level set function and capture the entire gap as one solid region.

Although the Random Walk Geodesic Active Contour idea is relatively simple in that it uses information about the distance between two clusters, it is extremely accurate. The model can virtually detect multiple gaps of any size in a 3D volume and propose a set of boundaries to bridge these gaps. However, the method is computationally expensive. For each identified section in the original image u, $O(n^3)$ distances are computed. Furthermore, the algorithm requires two three-dimensional segmentations. Hence, in an attempt to reduce the total number of segmentations to one, we investigate another segmentation approach that is based on the active contour without edges model derived by Chan and Vese in [6].

Algorithm 5 Random Walk Geodesic Active Contours

```
Require: The user must specify the values of the weights $w_1$, $w_2$, $w_3$, and $w_4$, and the
    probabilities $p$, $p_1$, $p_2$, and $p_3$. Typically, $w_1 = w_2$, $w_3 > w_1$, and $w_4 \gg w_3$; $0.25 < p \leq$
    $0.5$, $p_1 > 0.5 > p_2 \geq p_3$, and $p_1 + p_2 + p_3 = 1$.
 1: Divide the original image $u$ into sections based on voxel intensity values and location.
    The clustered image is called $u_c$ and contains $K$ sections labeled $C_k$
 2: for $k = 1 \to K$ do
 3:   Determine the set of points $S_k \subset u$ that should be connected to $C_k$
 4:   Identify a possible gap region $R_k$ which is bounded by $C_k$ and $S_k$
 5:   Initialize $L$ to be an $n \times n \times n$ matrix of zeros
 6:   if $S_k$ is above $C_k$ then
 7:     for all points $p$ on the bottom face of $S_k$ do
 8:       $X \leftarrow$ RandomWalkDown$(N, p)$
 9:       if $X(N, :) \in C_k$ then
10:         $L(x, y, z) \leftarrow 1 \quad \forall (x, y, z) \in X$
11:       end if
12:     end for
13:   else
14:     for all points $p$ on the top face of $S_k$ do
15:       $X \leftarrow$ RandomWalkUp$(N, p)$
16:       if $X(N, :) \in C_k$ then
17:         $L(x, y, z) \leftarrow 1 \quad \forall (x, y, z) \in X$
18:       end if
19:     end for
20:   end if
21: end for
22: Initialize $\phi_0$ as a signed distance function
23: $m \leftarrow 0$
24: repeat
25:   for each voxel location $(i, j, k)$ do
26:     if $L(i, j, k) = 1$ then
27:       $c^m_{ijk} \leftarrow -\kappa^m_{ijk}$
28:     else
29:       $c^m_{ijk} \leftarrow 0$
30:     end if
31:     $F^m_{ijk} = -\big(g(\nabla u)_{ijk}(c^m_{ijk} + \kappa^m_{ijk}) + (\nabla g(\nabla u))_{ijk} \cdot \frac{\nabla \phi^m_{ijk}}{|\nabla \phi^m_{ijk}|}\big)$
32:     $\phi^{m+1}_{ijk} = \phi^m_{ijk} - \Delta t\big[\max(F_{ijk}, 0)\nabla^+ + \min(F_{ijk}, 0)\nabla^-\big]$
33:   end for
34:   $m \leftarrow m + 1$
35: until $|\phi^{m+1} \geq 0| = |\phi^m \geq 0|$
36: Return $\phi^{m+1}$
```

3.2 3D-2D Level Set Segmentation

The Chan-Vese model detects objects in an image by comparing the intensity value of each voxel (x, y, z) with the mean intensity values inside and outside the embedding function. However, a point inside the gap of an incomplete cell path has the same intensity value as the background, as opposed to the intensity values within the cell tubes. Therefore, the level set surface will move away from the gap and simply capture the visible cells, as illustrated in Figure 3.3(a). An important property of active contours without edges is that if $c_1 = c_2$, that is the average intensity values inside and outside the embedding function ϕ are equal, then the fitting term in (2.10) is minimized for that particular ϕ. Furthermore, if $\nu = 0$, so that there is no volume constraint on the energy functional, then the level set surface will not evolve since the given ϕ is already an optimal solution. We use this particular property in our 3D-2D level set segmentation method to avoid the undesirable motion of the level set surface moving away from the gap.

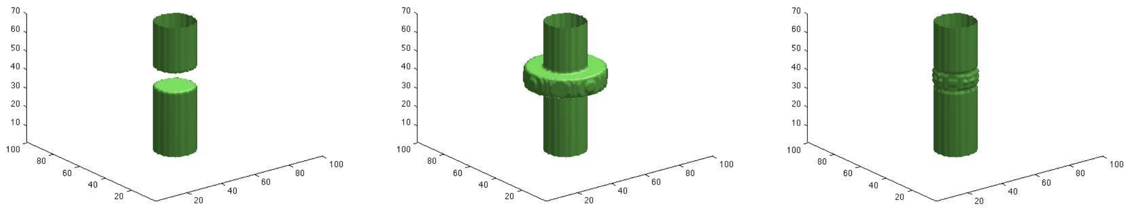
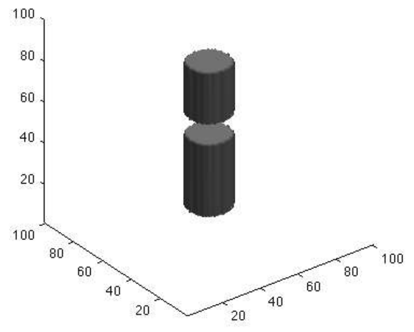

(a) Chan-Vese (b) 3D-2D with $\mu = \nu = 0$ (c) 3D-2D with $\mu,\ \nu \neq 0$

Figure 3.3: Segmentation of a cylinder with a gap of height 10 voxels

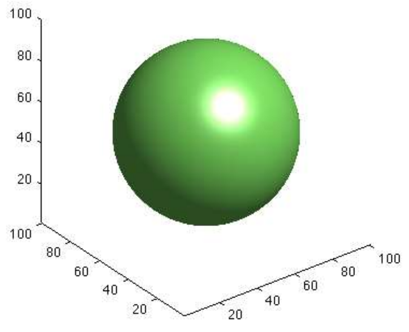
In order to detect gaps in the z-direction, we propose that the comparison of intensity values takes place at a two-dimensional slice instead of the three-dimensional volume. More precisely, suppose we want to classify the voxel (x, y, z) as a point inside or outside the level set surface ϕ. Then we consider the image frame corresponding to the z position, say u^z, and project ϕ onto frame z to obtain a two-dimensional level set function:

$$\phi^z(x, y) = \phi(x, y, z).$$

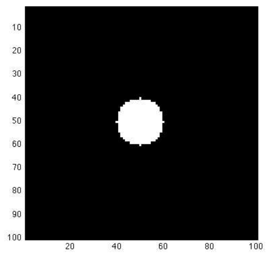
This idea is illustrated in Figure 3.4. The image u is a cylinder that exhibits a gap between the slices $z = 50$ and $z = 60$ (Figure 3.4(a)). The zero level set surface of the initial embedding function ϕ_0 is a sphere that encompasses the entire cylinder, including



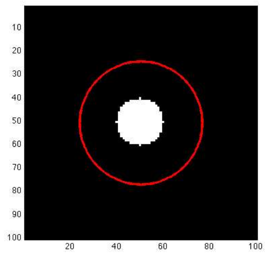
(a) Initial 3D volume u



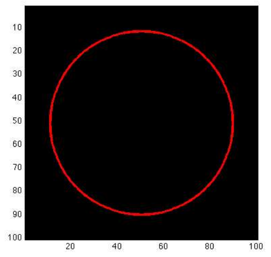
(b) Level set surface, $\phi_0(x,y,z) = 0$



(c) Frame 80 of u, u^{80}



(d) $\phi_0^{80}(x,y) = 0$



(e) Inside the gap, $\phi_0^{55}(x,y) = 0$

Figure 3.4: Projection of the initial embedding function onto two frames of an image

the gap, as shown in Figure 3.4(b). At any point on the eightieth slice of u, denoted u^{80} (Figure 3.4(c)), we consider the two-dimensional level set function $\phi_0^{80}(x,y) = \phi_0(x,y,80)$ for which the zero level set curve is shown in Figure 3.4(d).

Let c_1^z and c_2^z be the average intensity values of u^z inside and outside ϕ^z respectively. Then we replace c_1 and c_2 in the Euler-Lagrange equation (2.12) with c_1^z and c_2^z. Our proposed partial differential equation for ϕ is given by

$$\frac{\partial \phi}{\partial t} - \delta(\phi)\left[\mu \cdot \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - \nu - \lambda_1(u - c_1^z)^2 + \lambda_2(u - c_2^z)^2\right] = 0. \tag{3.3}$$

This equation evolves the 3D level set function ϕ by performing a 2D level set segmentation on each frame. If a cell is present in an image frame z, as in Figure 3.4(c), the λ_1 and λ_2 terms in (3.3) will attract ϕ^z and hence ϕ towards the cell boundary. This is similar to the two-dimensional Chan-Vese segmentation model. However, for an image frame within the gap of the volume such as in Figure 3.4(e), $c_1^z = c_2^z$. Therefore, as described previously, in the case where $\nu = 0$, the zero level set curve will be identical to the initial one. This is observed in Figure 3.3(b) where the gap is captured by the segmentation, but it is not an accurate representation of the missing cells. It is important to note that the 3D-2D approach is not the same as segmenting each image frame individually. The latter would not be able to bridge the gap and would simply produce the same results as a three-dimensional image segmentation. In our method, each projected level set function ϕ^z locates the boundaries of the objects in its own image frame while maintaining information about the entire 3D level set function ϕ. In fact, this information is maintained by the regularization terms, μ and ν. For illustration purposes, consider a unique incomplete cell path with one gap. Suppose the cell is visible in frame z_1 but non-existent in the next frame $z_1 + 1$. Then the projected level set function ϕ^{z_1} is drastically different from ϕ^{z_1+1} because the first contour captures the cell in the 2D image whereas the second is given by the initial level set function. This large difference in contours induces a large mean curvature in the zero level set of ϕ. The regularization terms are meant to minimize this effect by evolving ϕ^{z_1+1} such that the contour will be close to the cell boundary given by ϕ^{z_1}. Hence, with an appropriate choice of μ and ν, the 3D-2D method will correctly capture the gap as in Figure 3.3(c).

In summary, the λ_1 and λ_2 terms drive the segmentation of the two-dimensional image towards the existing objects in the frame, capturing the visible parts of the cell tube. The regularization terms then extend the surface by minimizing the mean curvature into the region where the cell disappears, creating a segmentation of the missing cell. This method does not rely on physically identifying a gap region, it simply manipulates the level set surface to evolve in a desired manner when there is a gap within the embedding function. In other words, the level set curve is not attracted towards the gap, instead, it is designed to replicate contours from image frames prior to and following the gap. Therefore, in order for this model to produce accurate results, the initial embedding function ϕ_0 must completely surround the gap.

To handle the case of an image containing many cell tubes with multiple gaps, we suggest dividing the 3D volume into sections and performing the 3D-2D level set segmentation on each of these smaller images separately. Furthermore, if $u_1^z = u_2^z$, $u_1 = u_2$, and $u_1^z = u_1$ for some $z \in [1, n]$, then the entire image contains no objects, so we return $\phi(x, y, z) < 0 \, \forall (x, y, z) \in u$.

Chapter 4

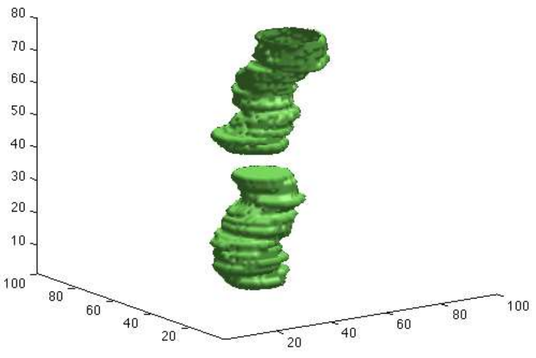
Results

In this chapter, we present the results of the algorithms described in Chapter 3. All of the following cell images are fluorescent microscopy images of live C2C12 cells obtained from the Department of Medicine and Human Genetics at McGill University. The original image size is 512×512, but for illustration purposes, only a 100×100 section is shown. We present the results obtained from three different datasets, each consisting of eighty frames. First we test both methods on a 3D volume image containing one cell. Then we use a dataset in which cell division is observed to discuss the effect of gap sizes on the segmentation results. Finally we show the results for a more complex dataset with multiple cells.

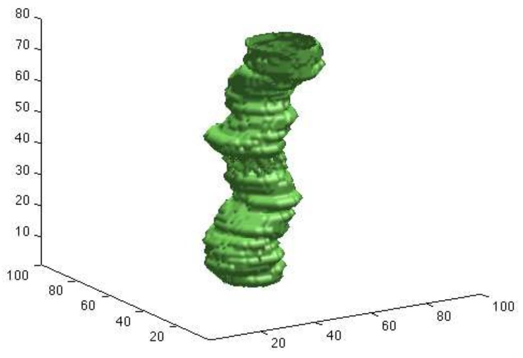
The parameters for the random walk geodesic algorithm are set as follows, $p = 0.45$, $p_1 = 0.8$, $p_2 = p_3 = 0.1$, $w_1 = w_2 = 1$, $w_3 = 2$, and $w_4 = 100$. The height of the largest gap we wish to be capable of identifying, h, depends on the experiment we are running. Unless otherwise specified, we choose $h = 10$, which is the average predicted gap size. For the 3D-2D level set method, the values of μ and ν depend on the dataset, however, we always use $\lambda_1 = \lambda_2 = 1000$ and a time step of $\Delta t = 0.00009$

4.1 Images Containing One Cell

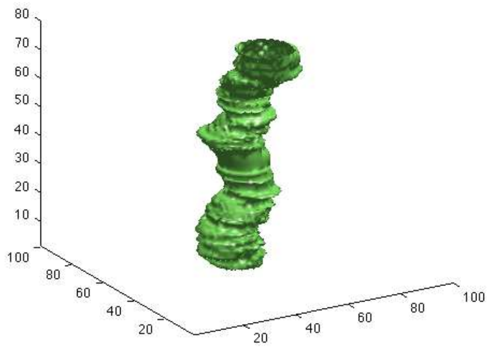
Figure 4.1 shows the results of the random walk geodesic and the 3D-2D level set algorithms applied to an incomplete cell path containing one gap of height 8 voxels. A complete image dataset with the cell visible in each frame is taken as the ground truth. We then manually remove the cell from frames $z = 36$ to $z = 43$ to simulate the effect of a cell disappearing and reappearing from the frame of interest. For this experiment, we choose the 3D-2D



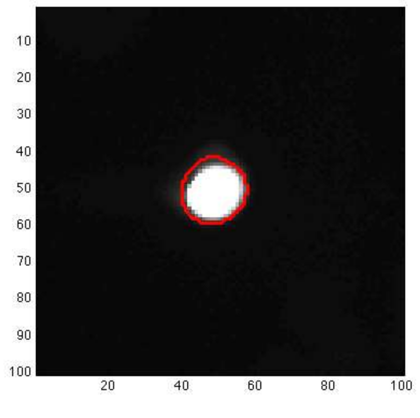
(a) Regular level set



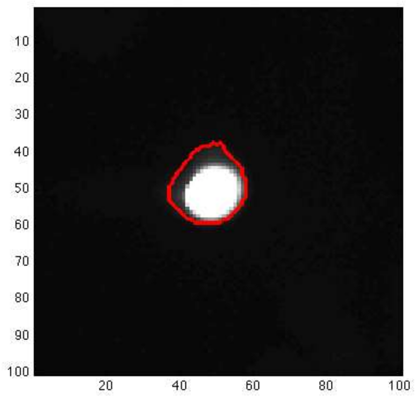
(b) Random walk geodesic



(c) 3D-2D level set



(d) Random walk geodesic, $z = 36$



(e) 3D-2D level set, $z = 36$

Figure 4.1: Segmentation results of images containing one cell

parameters to be $\mu = 0.1$ and $\nu = 1$. The gap is visible in Figure 4.1(a) which corresponds to an ordinary level set segmentation result, however, both the random walk and the 3D-2D level set methods reconstruct the incomplete cell path, as illustrated in Figures 4.1(b) and 4.1(c). In Figures 4.1(d) and 4.1(e), we project the predicted segmentation surfaces onto frame 36 of the ground truth dataset; this corresponds to the actual cell. Observe that the curves agree very well with the missing cell, however, in this particular case, the random walk geodesic method gives a better approximation of the cell location than the 3D-2D level set method. It is important to note that the 3D-2D level set result is also extremely accurate, even though the contour captures the glow of the cell from the microscope. In fact, this result is expected of a level set segmentation in which no measures are taken to smooth the image.

For this dataset, we compute the accuracy of the random walk geodesic segmentation results for a gap size of eight voxels. Ten such cases are analyzed and recorded in Table 4.1 for a total of eighty image segmentation results. Similarly, the accuracy of the 3D-2D level set method is recorded in Table 4.2. Accuracy is measured as $|A \cap B|/|A|$ where A denotes the segmentation result and B the ground truth. That is, in a given frame, we determine the percentage of pixels inside a 35×35 square containing the original cell that are correctly classified by the level set function. The frame numbers in both tables correspond to the distance in the z-direction between the frame of interest and the beginning of the gap. For example, in Case 1, we manually remove the cell from frames $z = 14$ to $z = 21$; then the first entry indicates the accuracy of the segmentation at slice 14, the second indicates the accuracy at slice 15, and so forth.

The accuracy is consistently high for the random walk geodesic algorithm and does not seem to depend on the distance between the missing cell frame and the known positions. On the other hand, for the 3D-2D level set segmentation algorithm, the accuracy is higher at the extremities of the gap and becomes worse in the middle since the curve is further away from the known positions. However, on average, this method captures most of the missing cell with a mean accuracy of 83.78% . For comparison purposes, the mean accuracy of both methods at each frame is plotted in Figure 4.2. For this particular dataset, the random walk algorithm is comparable to the 3D-2D method at the extremities of the gap but seems to dominate in the centre.

Figure 4.3 illustrates that both methods are capable of detecting multiple gaps of different sizes within one cell tube. Given the unicellular complete image dataset, we remove the cell from frames $z = 22$ to $z = 29$ and from $z = 55$ to $z = 63$, as shown in Figure 4.3(a). Once again, we observe an appropriate reconstruction of the incomplete cell path.

	Frame							
	1	2	3	4	5	6	7	8
Case 1	0.8606	0.8775	0.8650	0.8700	0.8812	0.8694	0.8669	0.8619
Case 2	0.8794	0.8875	0.8856	0.8894	0.9094	0.8894	0.8900	0.8881
Case 3	0.8950	0.8925	0.8950	0.8925	0.9031	0.8987	0.8956	0.8912
Case 4	0.8756	0.8819	0.8838	0.8775	0.8819	0.8769	0.8731	0.8756
Case 5	0.8869	0.8825	0.8800	0.8888	0.8794	0.8888	0.8806	0.8912
Case 6	0.8975	0.8888	0.8944	0.8987	0.9012	0.8888	0.8894	0.8900
Case 7	0.8931	0.9069	0.9175	0.9006	0.9031	0.9000	0.8919	0.8925
Case 8	0.8781	0.8781	0.8806	0.8869	0.8812	0.8769	0.8719	0.8781
Case 9	0.8781	0.8781	0.8806	0.8869	0.8812	0.8769	0.8719	0.8781
Case 10	0.8688	0.8619	0.8750	0.8725	0.8800	0.8900	0.9100	0.8981
Mean	0.8813	0.8836	0.8858	0.8864	0.8902	0.8856	0.8841	0.8845

Table 4.1: Accuracy of random walk geodesic segmentation results for 10 testing cases, each with a gap of 8 voxels

	Frame							
	1	2	3	4	5	6	7	8
Case 1	0.9500	0.9063	0.9375	0.6500	0.6625	0.8750	0.8936	0.9188
Case 2	0.9500	0.9000	0.8438	0.6750	0.5375	0.7750	0.9250	0.9500
Case 3	0.9375	0.8750	0.9250	0.8750	0.8812	0.9188	0.9438	0.8938
Case 4	0.9250	0.9375	0.9250	0.6875	0.6813	0.7813	0.7875	0.8750
Case 5	0.9313	0.8313	0.9375	0.6688	0.6563	0.7063	0.8250	0.8313
Case 6	0.8938	0.7813	0.6438	0.6375	0.7313	0.9000	0.8688	0.8750
Case 7	0.8875	0.9500	0.9375	0.8688	0.7750	0.8438	0.9938	0.9000
Case 8	0.8500	0.9000	0.8938	0.7625	0.8000	0.9125	0.8375	0.8438
Case 9	0.8438	0.8250	0.7000	0.7312	0.8375	0.9063	0.9375	0.9187
Case 10	0.8125	0.8688	0.7750	0.5938	0.7688	0.8375	0.8938	0.8988
Mean	0.8981	0.8775	0.8519	0.7150	0.7331	0.8457	0.8906	0.8905

Table 4.2: Accuracy of 3D-2D level set segmentation results for 10 testing cases, each with a gap of 8 voxels

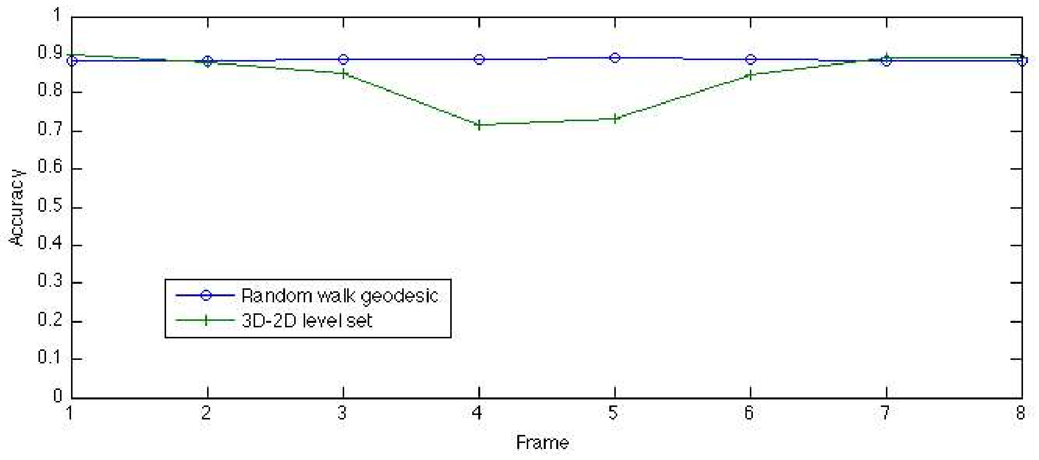


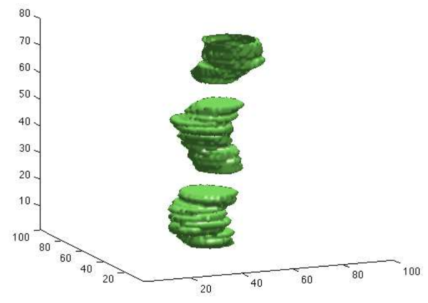
Figure 4.2: Comparison of the mean accuracy between both methods, for 10 testing cases, each with a gap of 8 voxels

4.2 Effect of Gap Sizes

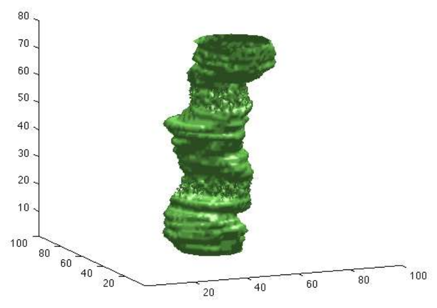
For simple datasets such as the unicellular example from section 4.1, the prism in Figure 2.4, or the cylinder in Figure 3.3, the gap size has no effect on the final random walk geodesic level set segmentation. Simply setting h to be large enough allows for the detection of gaps of any size. For example, we show the random walk geodesic level set results applied to an incomplete cell path in which the cell disappears from six, eleven, and sixteen frames. We set $h = 16$ and record the results in Figure 4.4. In each case, the segmentation contour captures the gap quite accurately. For comparison purposes, the complete cell path is shown in Figure 1.2(a).

However, for a less consistent dataset in which the different clusters are not as compact, such as in Figure 4.5 where the motion of the cell in the lower half of the volume spans a large portion of the image frame, the results obtained from the random walk geodesic active contours algorithm are not quite as accurate. We perform the next experiment on both of the image reconstruction approaches introduced in this paper. As in Section 4.1 we set the 3D-2D parameters to be $\mu = 0.1$ and $\nu = 1$. The ground truth data consists of a sequence of images in which the cell initially exhibits an abnormally rapid movement towards the centre of the frame, and then undergoes cell division. We manually remove a cell from six, eleven, and sixteen frames, respectively, as shown in Figure 4.5.

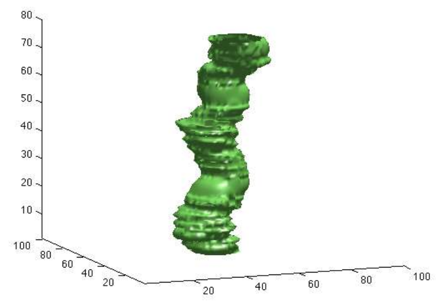
First, the random walk geodesic segmentation is applied to reconstruct the incomplete



(a) Regular level set

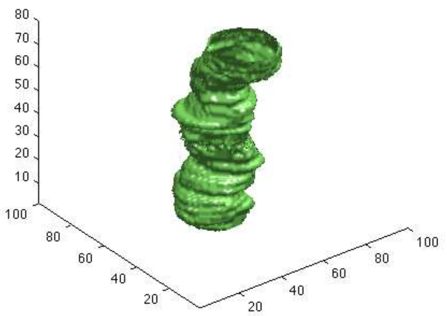


(b) Random walk geodesic

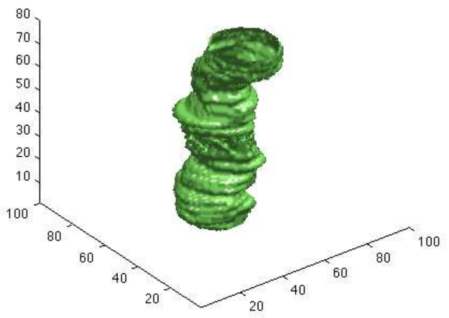


(c) 3D-2D level set

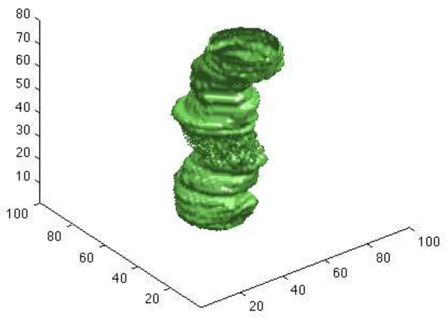
Figure 4.3: Segmentation results of a cell tube with two gaps



(a) Gap size = 6 voxels

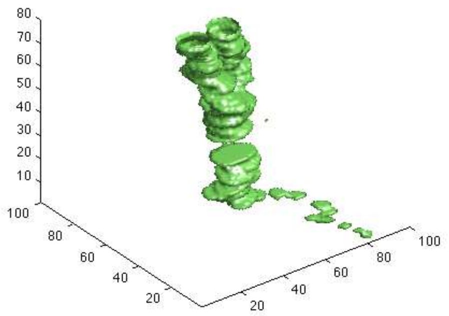


(b) Gap size = 11 voxels

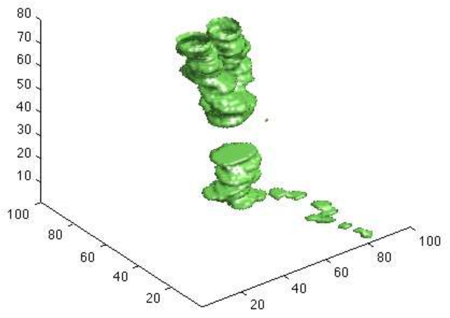


(c) Gap size = 16 voxels

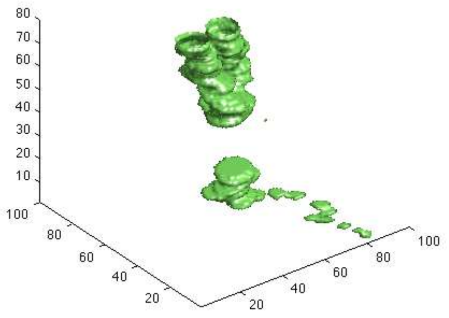
Figure 4.4: Random walk geodesic reconstruction of incomplete cell paths with different gap sizes



(a) Gap size = 6 voxels



(b) Gap size = 11 voxels



(c) Gap size = 16 voxels

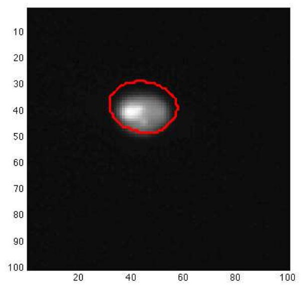
Figure 4.5: Incomplete cell paths with different gap sizes in which the cell undergoes division

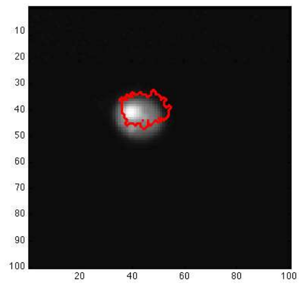
cell paths. For each gap size, four image frames in which a cell is missing, along with the resulting contours, are shown in Figure 4.6. In each row, the first and the last images correspond to the extremities of the gap, the other two represent the middle frames. The results for the small gap size in Figures 4.6(a) and 4.6(d) are almost one hundred percent accurate, however in every other image, the active contour seems to underestimate the area of the cell. Furthermore, as the gap size increases, the area of the predicted cell region decreases. This result is explained by the unusual behaviour observed in the bottom half of the 3D image volume. The rapid movement of the cell in the first few frames is not taken into account by the model since the probabilities in the transition matrix of the random walk only depend on the previous state. Hence, in order to resolve this issue, we suggest using the known positions of the visible cells in the incomplete cell path to propose a stochastic process that models the cell behaviour. The probabilities used in the random walk algorithm should then be derived from the stochastic process.

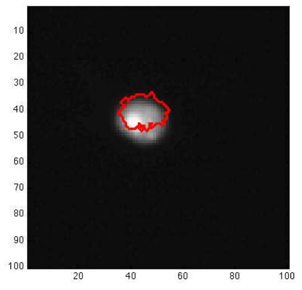
The results obtained when the 3D-2D segmentation is applied to reconstruct the incomplete cell path are shown in Figure 4.7. The organization of the images is the same as in Figure 4.6. For this method, the segmentation contour agrees very well with the missing cell. Note that the results are more accurate for smaller gap sizes and tend to get worse as the gap size increases. Also, as observed in the previous section, the predicted cell location is better at the extremities of the gap than in the middle.

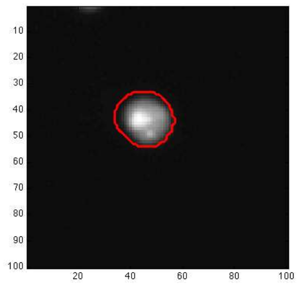
4.3 Images Containing Multiple Cells

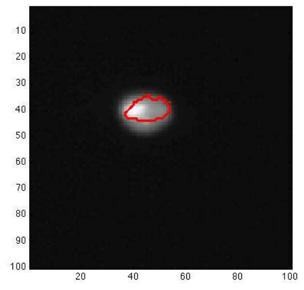
Finally, we present the segmentation results for a 3D image volume containing multiple cells and two gaps. A complete image dataset with two cells in each frame, one of which undergoes a cell division, is taken as the ground truth. We then manually remove a cell from frames $z = 16$ to $z = 22$ and a different one from frames $z = 60$ to $z = 66$, creating two gaps of height 7 voxels. For this experiment, we set $\mu = 0.02$ and $\nu = 2$. The gaps are visible in Figure 4.8(a), which corresponds to the regular level set segmentation result. As expected, both the random walk and the 3D-2D segmentation methods reconstruct the incomplete cell paths, which is illustrated in Figures 4.8(b) and 4.8(c). In the 2D images, we show the segmentation results at slices $z = 17$ and $z = 61$. The green curve represents the segmentation of the cells that are visible in the image frame, whereas the red curve represents the predicted location of the missing cell. We note that both methods capture the cells quite accurately, however, the results of the 3D-2D segmentation are smoother and slightly more accurate.

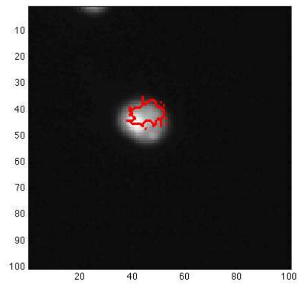

(a) Gap = 6, $z = 36$

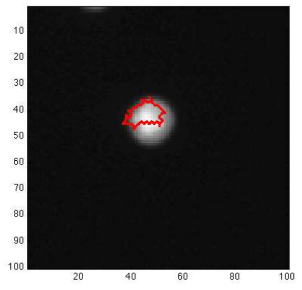

(b) Gap = 6, $z = 38$

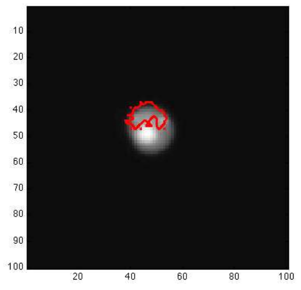

(c) Gap = 6, $z = 40$

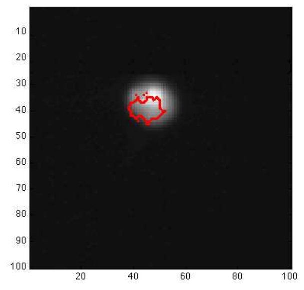

(d) Gap = 6, $z = 41$

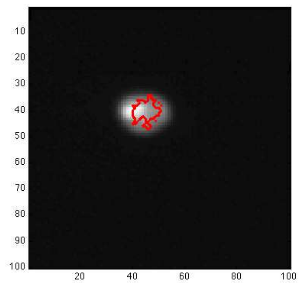

(e) Gap = 11, $z = 36$

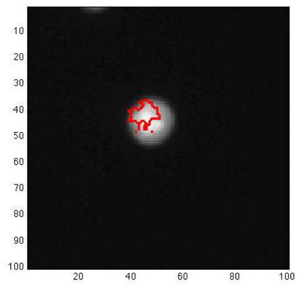

(f) Gap = 11, $z = 40$


(g) Gap = 11, $z = 43$


(h) Gap = 11, $z = 46$


(i) Gap = 16, $z = 31$


(j) Gap = 16, $z = 36$


(k) Gap = 16, $z = 43$

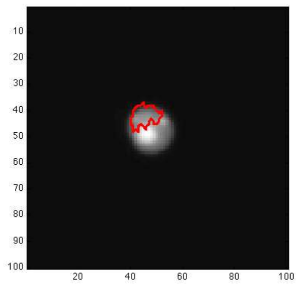
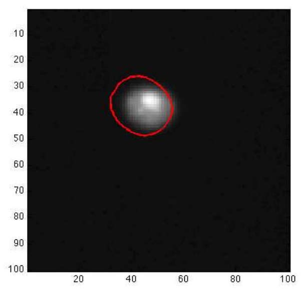
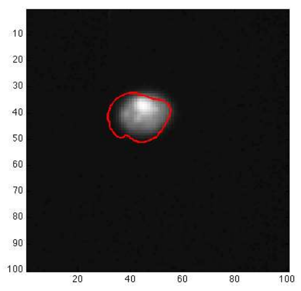
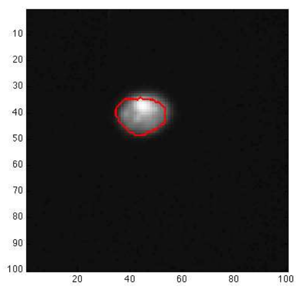
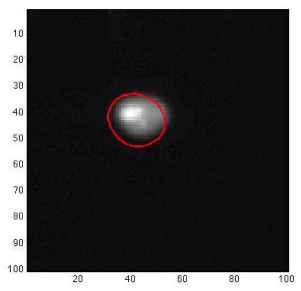

(l) Gap = 16, $z = 46$

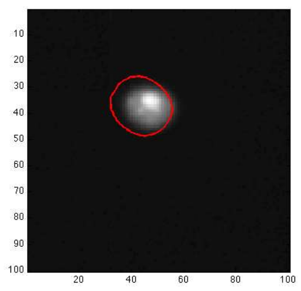
Figure 4.6: Effect of gap sizes on random walk geodesic active contours model

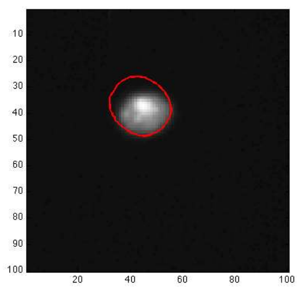

(a) Gap = 6, $z = 32$

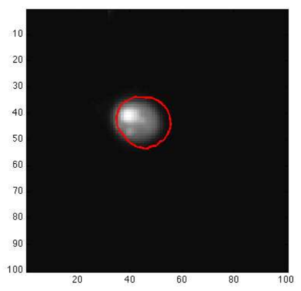

(b) Gap = 6, $z = 34$

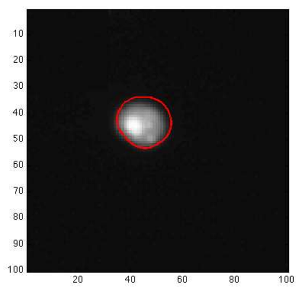

(c) Gap = 6, $z = 35$

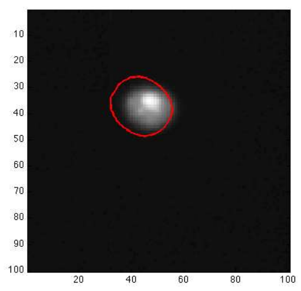

(d) Gap = 6, $z = 36$

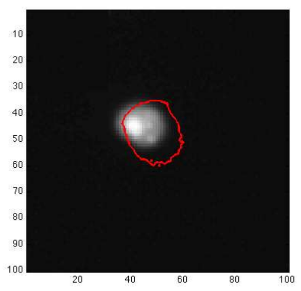

(e) Gap = 11, $z = 32$

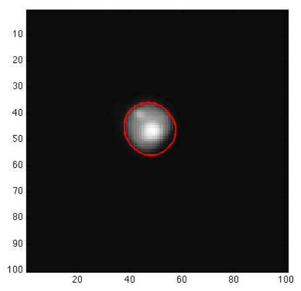

(f) Gap = 11, $z = 34$


(g) Gap = 11, $z = 38$


(h) Gap = 11, $z = 40$


(i) Gap = 16, $z = 32$


(j) Gap = 16, $z = 40$


(k) Gap = 16, $z = 44$

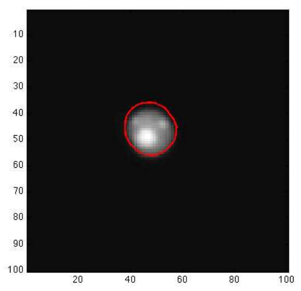
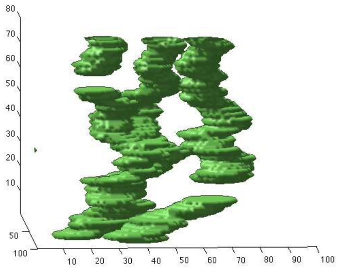
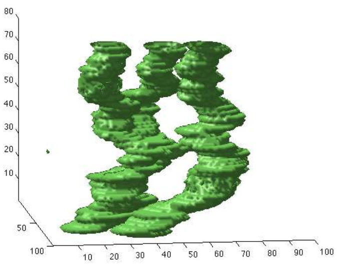
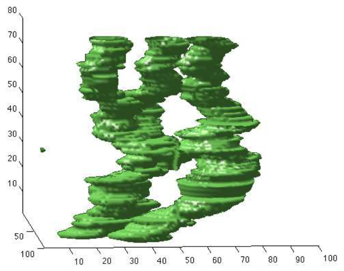
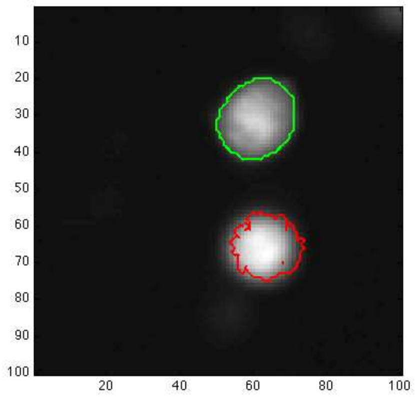

(l) Gap = 16, $z = 46$

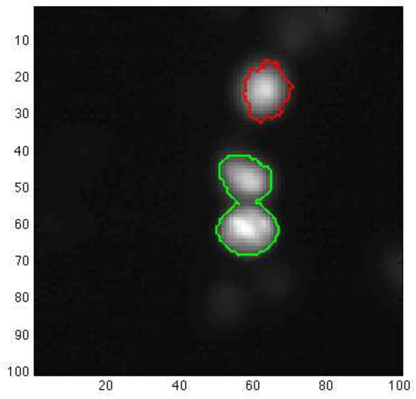
Figure 4.7: Effect of gap sizes on 3D-2D level set method

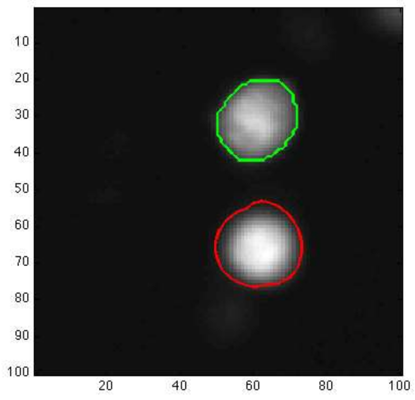

(a) Regular level set


(b) Random walk geodesic


(c) 3D-2D level set


(d) Random walk geodesic, $z = 17$


(e) Random walk geodesic, $z = 61$


(f) 3D-2D level set, $z = 17$

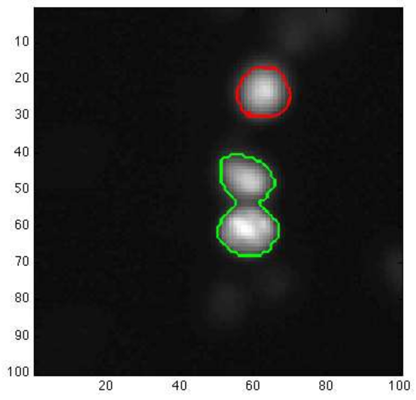

(g) 3D-2D level set, $z = 61$

Figure 4.8: Segmentation results of multiple cell tubes with two gaps

Chapter 5

Conclusion

This paper has presented a solution to one of the common problems that occur when tracking cells in fluorescent microscopy images. In particular, we proposed two different methods that used three-dimensional segmentation to reconstruct incomplete cell paths when some of the cells disappear and reappear in the image sequence obtained from the microscope.

The first model, called Random Walk Geodesic Active Contours, was a three step process that used information about distance and voxel intensity values to locate the gaps in the 3D image volume. Then the Metropolis algorithm from Markov Chain Monte Carlo methods was modified to propose a set of boundaries to bridge these gaps. However, the algorithm was computationally expensive and required two three-dimensional segmentations. Hence, we proposed a second model, called 3D-2D Level Set, which performed a two-dimensional segmentation in a three-dimensional framework. In this case, we enforced minimum mean curvature on the level set surface to capture the gaps in the 3D volume.

We showed that both methods were able to correctly reconstruct the incomplete cell paths on a number of different image datasets. However, the random walk geodesic model did not perform as well in a 3D volume in which the motion of the cell spanned a large portion of the image frame.

Possible future work includes deriving a stochastic process that models the cellular dynamics in the incomplete cell path, so that the accuracy of the random walk method is not affected by the behaviour of the cells in the 3D volume. Also, further testing on the robustness of the 3D-2D method should be performed. In particular, we should test the model on images in which the cell goes missing from the frame right before it undergoes a cell division.

References

[1] D. Adalsteinsson and A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118(2):269–277, may 1995.

[2] R. Ananthakrishnan and A. Ehrlicher. The forces behind cell movement. *International Journal of Biological Sciences*, 3:303–317, 2007.

[3] A. Bosnjak, G. Montilla, R. Villegas, and I. Jara. 3D segmentation with an application of level set-method using MRI volumes for image guided surgery. *IEEE EMBS*, pages 5263–5266, aug 2007.

[4] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours. *Numerische Mathematik*, 66:1–31, 1993.

[5] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

[6] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2), feb 2001.

[7] Shojaeddin Chenouri. Computational inference course notes, 2011.

[8] M. Kass, A. Witkin, and D. Tersopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[9] W. Kenong, D. Gauthier, and M.D. Levine. Live cell image segmentation. *IEEE Transactions on Biomedical Engineering*, 42(1):1–12, January 1995.

[10] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[11] G. Rabut and J. Ellenberg. Automatic real-time three-dimensional cell tracking by fluorescence microscopy. *Journal of Microscopy*, 216(2):131–137, November 2004.

[12] James A. Sethian. *Level Set Methods and Fast Marching Methods*, pages 3–13. Cambridge University Press, 1999.

[13] S. Tse, L. Bradbury, W.L. Wan, H. Djambazian, R. Sladek, and T. Hudson. A combined watershed and level set method for segmentation of brightfield cell images. February 2009.