5-2012

Adaptive Re-Segmentation Strategies for Accurate Bright Field Cell Tracking

Nare Hayrapetyan
Utah State University

ADAPTIVE RE-SEGMENTATION STRATEGIES FOR ACCURATE BRIGHT FIELD CELL TRACKING

by

Nare Hayrapetyan

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Nicholas Flann Major Professor	Dr. Daniel Watson Committee Member
Dr. Renée Bryce Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2012

ABSTRACT

Adaptive Re-Segmentation Strategies for Accurate Bright Field Cell Tracking

by

Nare Hayrapetyan, Master of Science

Utah State University, 2012

Major Professor: Dr. Nicholas Flann
Department: Computer Science

Understanding complex interactions in cellular systems requires accurate tracking of individual cells observed in microscopic image sequences and acquired from multi-day in vitro experiments. To be effective, methods must follow each cell through the whole experimental sequence to recognize significant phenotypic transitions, such as mitosis, chemotaxis, apoptosis, and cell/cell interactions, and to detect the effect of cell treatments. However, high accuracy long-range cell tracking is difficult because the collection and detection of cells in images is error-prone, and a single error in one frame can cause a tracked cell to be lost. Detection of cells is especially difficult when using bright field microscopy images wherein the contrast difference between the cells and the background is very low. This work introduces a new method that automatically identifies and then corrects tracking errors using a combination of combinatorial registration, flow constraints, and image segmentation repair.

(52 pages)

PUBLIC ABSTRACT

Adaptive Re-Segmentation Strategies for Accurate Bright Field Cell Tracking

Cell migration is central for many fundamental biological processes and development of multi-cellular organisms. The failure of cells to migrate or migration of cells to inappropriate locations during embryo development can result in life threatening consequences such as brain malfunctions. In adults, cell migration plays an important role in wound healing and immune responses. Failure in these processes can have dramatic medical implications and can lead to vascular diseases and tumor formation. Therefore, studying cell migration is critical to helping prevent and cure diseases.

Cell migration is usually studied by observing cells photographed through a microscope at regular time intervals. However, because of the sheer amount of data, this is a time-consuming and difficult task. There is an obvious need for automated approaches that can correctly identify and follow cells. Many such approaches exist, but they are very error-prone due to poor image quality and difficulty of the task. Here, we present an approach for automatically detecting and correcting those errors, which will increase the accuracy of automated methods.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Tracking cells from image sequences is critical for understanding multicellular phenomena such as wound repair [32], immune responses [8], embryonic development [2], and cancer progression [4]. This work introduces an enhanced tracking algorithm that iteratively improves cell identification and segmentation, based on errors identified by high-performance multi-frame alignment methods. The method is validated in two key biological systems. First is the epithelial to mesenchymal transition (EMT) in cancer cells, wherein EMT is a series of events in which cell/cell interactions are altered and cell adhesion is lost to release epithelial cells from the surrounding tissue [21]. Studying EMT evaluates metastatic potential which is directly correlated with cancerous cells spreading from one organ to another. The second biological system we used to validate our method consists of HEC human embryonic kidney cells that are an important model for cell differentiation and motility.

Cell tracking is defined as finding the correspondence between cells in consecutive images to gather cell trajectories. Current cell tracking methods [10, 20] are most effective using high contrast and noise free images, fluorescently-labeled cell nuclei, and distinct cell boundaries. However, ideal conditions rarely exist. Most recent work aimed at addressing said problems apply two alternative approaches. The first is a segmentation-based method that initially detects cells in each image through segmentation and then applies alignment to match cell identity over adjacent frames [31]. The second method utilizes deformable models to register objects in adjacent images based on physically realistic shape and movement changes [15]. This thesis extends and improves the segmentation-based approach under low-contrast and noisy conditions.

Fluorescently tagging cells using the jellyfish green fluorescent protein (GFP) leads to robust identification of each cell during segmentation and subsequent high accuracy track-

ing. While GFP tagging is widespread, there are disadvantages when applying the method for cell tracking because cells can suffer phototoxicity due to the repeated application of high-energy light. Such light can disrupt the cell behavior through stress, shorten life and potentially confound the experimental results [1, 24]. Significantly, a requirement for GFP labeling adds a step before a new cell line can be tracked, thus making it difficult to use cell tracking in a clinical setting.

The alternative is to use bright field microscopy, the original and the simplest microscopy technique, wherein cells are illuminated with white light from below. However, using only bright field imaging of unstained cells presents a challenging cell detection problem because of lack of contrast and difficulty in locating cell centers, particularly when cells are tightly packed. Bright field imaging, while eliminating phototoxicity, leads to an excess of segmentation errors that can lower tracking accuracy to unacceptable levels.

We seek to remedy the disadvantages and harness the experimental advantages of bright field microscopy of living cells by automatically correcting segmentation errors detected through use of topological alignment tracking algorithms. Initially, the tracking process proceeds from segmentation to tracking, but rather than terminating, re-segmentations are performed on targeted individual objects in specific frames. The resulting segmentation corrections are fed back to the tracking algorithm, with the process iterating until residual errors are minimized.

This work presents a solution to the following problem: given a sequence of time-sampled bright field microscopy images acquired from in vitro live cell experiments, find and correct segmentation and tracking errors, such that the final output is a set of tracked and segmented cells with reduced number of errors, suitable for further processing such as morphological analysis, mitosis characterization, or florescent reporter extraction.

The experimental study was performed using a series of time-lapse images from two different cell types. The first data set is HMLER human breast cancer cells sampled with 324 images, taken at 5-minute intervals. The second data set is human embryonic kidney cells (HEK 293T) sampled with 193 images, taken at 5-minute intervals. Both data sets

were obtained using a Leica DM6000 microscope with each image containing 30 z-stack frames each separated by 10μm, with resolution 1024×1024 pixels.

CHAPTER 2

PREVIOUS WORK–LITERATURE REVIEW

Many methods have been developed for automatically detecting and tracking cells, each suggesting different approaches to overcoming challenges, such as poor image quality, high density of cell populations, motion, cells entering and leaving the image, cells merging and splitting, and cells touching each other with low image contrast. Following is an overview of recent cell segmentation and tracking algorithms.

2.1 Segmentation

Segmentation is the process of detecting individual cells on an image. As a result, a new image is produced with a monotone background containing contiguous regions of uniquely labeled pixels representing cells. The simplest proposed approach to segmentation is referred to as thresholding. It requires a single threshold parameter based on the histogram intensity of the image that is used to extract objects from the background. Pixels with values lower than the threshold value are labeled as background, and pixels with higher values are labeled as cells [14,16]. To further automatize the thresholding algorithm, [18] proposes an approach for automatically finding an optimal threshold value. The first step is to divide the pixels of the image into two classes - background and cells. The optimal threshold is then defined as the threshold that maximizes the separation between those classes so that their combined spread is minimal. Although really easy to implement, thresholding does not work well for clustered or overlapping cells nor for images with less contrast and variation in intensity such as bright field and phase-contrast microscopy imaging [31].

Another popular approach to segmentation is the watershed transformation. Here, the grayscale image is considered a topographic relief wherein the gray level of each pixel is its elevation. Vincent and Soille [27] define watersheds as the dividing lines of catchment basins.

A catchment basin is a set of pixels that have a descending path to a regional minimum. That is, if water falls on the pixel, it will flow down along the relief and reach the minimum. The segmentation occurs when the catchment basins are flooded from their minima which subdivides the image into regions. To efficiently implement the watershed method, [27] first sorts the pixels by their gray levels thus giving direct access to the pixels at any level. Then, the basins are flooded by a breadth-first scanning of each threshold level. The watershed method is effective in separating clustered cells but is sensitive to noise and tends to produce over-segmented results [26, 29]. Nevertheless, by applying some post-processing techniques, it is possible to obtain accurate results using this method. Some approaches include a rule-based merging of over-segmented regions [28] and a marker-controlled technique that corrects the overlapped regions [30].

Recently, another segmentation method called deformable models has been gaining interest [14]. In deformable models, objects are represented by closed contours or zero level-sets (a mathematical set in which the function takes on a value of a given constant). These contours, driven by internal and external forces, iteratively evolve, minimizing a predefined cost function until they converge to the boundaries of cells [29]. The cost function usually depends on image-dependent terms like intensity and gradients or image-independent terms like penalty on boundary length or curvature and similarity to a reference shape [31]. Since deformable models utilize both prior biological knowledge and constraints derived from the image, they can be easily adapted to specific applications [16].

Dufour et al. [7] present a segmentation method based on the deformable models approach. It seeks to segment the images based on an energy functional that has two data attachment terms: one penalizes the difference of intensity inside and outside the contour boundary, and the other penalizes the total boundary length. The method uses multiple level set functions instead of one, since single level sets are unable to identify a correct cell count when cells touch, unless cells have significantly different intensities. Therefore, each cell is represented by its own level set that penalizes the overlap with other cells. This prevents merges between cells but is not enough to detect where the boundary lies. Thus,

a constraint that requires cell volumes to remain constant is applied to guide the contours towards the boundaries. Due to strong constraints and optimizations, however, this method works well only with sphere-like convex cells [29].

Yan et al. [29] propose a more robust approach. It consists of two parts - segmentation of nuclei using a modified watershed method and using the segmented nuclei as initialization for deformable model segmentation of cells. First, to extract the nuclei, we use Otsu's thresholding method [18], which utilizes the contrast between the nuclei and the background. However, some cells are too close together, and they do not get separated using thresholding. A higher threshold would work better but would over-split many cells. To fix the problem, an enhanced watershed algorithm is developed that segments the image by finding a path from each pixel to the local minimum so each segment consists of all points whose paths of gradient descent end at the same local minimum. To avoid over-segmentation, a flood level threshold is calculated that does not allow overspilling into neighboring regions. The segmentation algorithm then combines regions whose depth falls below the flood level. The second step seeks to establish boundaries between cells. We formulate an energy functional based on a new interaction model that is minimized using a multiphase level set method. The interaction method consists of a set of rules that do not allow the contours to cross each other and the same point to belong to different cells. The model assigns points to cells by minimizing the formulated energy functional. The value of the function at each point is computed as the Euclidean distance to the nearest point on the boundary. A geodesic length measure [3] is used to snap the boundaries to image edges.

All above described methods are optimized for flourescently dyed cells since they need to utilize the contrast and intensity difference between the cells and the background. Unlike in fluorescence images, the cells that have not been treated with dyes are not consistently brighter than the background. However, there are advantages to using nonfluorescent microscopy, such as easier imaging setup [24] and the ability to observe cells for a long period of time with little alterations of cell physiology [31]. Korzynska et al. [11] propose an approach for segmenting bright field microscopy image sequences. The method is semi-automatic and

requires manual selection of parameters. The first step of the algorithm calculates the mean and the standard deviation of the gray level variation for each texture block and assigns it as part of a cell or the background by comparing the values to the mean and the standard deviation of values of a manually selected background reference region. The next step uses image gradient to determine the position of the cell contour. One-dimensional subspaces called profile lines are placed radially from the cell center and normalized cumulatively. The Prewitt edge operator function is used to find the points along the cell contours. Finally, the contour is formed from those points using spline interpolation. The proposed method can successfully segment bright field images and can cope with noisy images. However, it is not fully automatic and tends to produce over-segmentation errors.

Another approach for segmenting bright field images is described in [24]. The algorithm works with several focal planes that form a stack of 2D images in the z-dimension, called a z-projection stack. Since the method uses bright field microscopy images and the contrast between the cells and the background is not very high, an approach is developed to increase the contrast and make the detection of cells easier. This approach calculates the difference of intensities of corresponding pixels in the z-direction in the stack. The standard deviation of intensities is calculated for each pixel and projected on a 2D image. This way, the variability of cell intensities is increased as opposed to the intensity of the background, so the resulting image has a dark background with easily detected bright cells. The method is not computationally intensive and works well for images that have background intensity variations and low contrast.

2.2 Tracking

Tracking is the cell association step that consists of establishing correspondence between individual cells in successive images to obtain cell trajectories. The existing approaches for cell tracking can be roughly divided into two categories - separate segmentation and association and model-based contour propagation [31]. Tracking methods in the first group use one of the segmentation methods discussed above to obtain cell segments, which are then linked using different approaches. Methods for linking cell segments in successive frames

are further categorized into local and global methods [15]. The local methods link cells frame-by-frame, while global methods consider many frames at once. Linking is usually performed by assigning weights (scores) to cells based on proximity, overlap, size, intensity, as well as other measures of image and cell similarity. More features usually mean lower ambiguity, but they can also become restrictive as cells change shape, move around, split, enter and leave the frames [16].

One of the most accurate solutions to the tracking problem is the multiple-hypothesis tracking (MHT) method presented in [22]. In MHT, hypotheses (cell paths through the frames) are constructed for every cell in every frame. As new frames are processed, new hypotheses are formed, and compatibility constraints that do not allow paths to share cells are applied to ensure near-correct tracking. Next, the probability of each hypothesis that is part of a potential track is calculated, and the hypothesis matrix is simplified by removing the unlikely associations until only one hypothesis remains for every cell. This algorithm, however, is very inefficient when tracking many cells, so many heuristics have been proposed to approximate MHT solutions. Mostly, these are greedy algorithms that seek to approximate the solution by choosing locally optimal associations.

Chen et al. [4] suggest a method to preform local frame-by-frame tracking. Their method detects a correspondence between cells in frame t and frame t+1 by computing Euclidean distances between centers of gravity of cells. Four types of matching are possible: one cell matches another in t+1, no cell in t+1 matches a cell in t (disappear), one cell matches more than one in t+1 (split), and more than one in t match one in t+1 (merge). The cases are ranked by smallest distance first. Ambiguous correspondence between touching cells is recorded and resolved at a later frame when cells move away from each other.

A more extensive framework is developed in [20] that handles cell behaviors, such as splitting, merging, entering and leaving and moving in the frame. The framework consists of a weighted bipartite graph that matches cells in one frame to another by minimizing the sum of the values of the edges. The values are based on cell features, behaviors, and movement. The graph is extended to enable the modeling of splitting, merging, disappearing, and

appearing cells by adding extra cells and edges that can merge and split in the middle of another edge to make the count of cells the same in both frames. The framework is easily extensible and does not require parameter tuning.

Mosig et al. [17] propose to minimize the over and under-segmentation errors by mapping a set of segments to another set of segments in the next frame maximizing the overlap between the frames. The algorithm assigns weights to segment sets, and the sets that have high overlap receive a high weigh score and vice-versa. The sets that receive a score close to 1 are considered one cell. The overlap and score information gathered in one frame is then transferred to the next and previous frames. The score and cell/cell matching constraint problem is defined as a linear programming problem that computes a maximum-edge-weighted bipartite matching. Many-to-many mappings used in this algorithm make it easier to handle events such as cell division.

Jaqaman et al. [10] combine a greedy approach of matching cells between consecutive frames and a global optimization approach by connecting the resulting tracked segments to close gaps from temporary disappearance and resolve cell merging and splitting. In the frame-to-frame cell linking step, there are three types of assignments possible: 1) a cell in frame t can link to a cell in frame t+1, 2) a cell in frame t can link to nothing in t+1 (disappear), 3) or a cell in t+1 can link to nothing in frame t (appear). In the next step, there are six types of assignments possible: 1) the end of a segment in frame t can link to a start of a segment in frame t+1, 2) the end of a segment in frame t can link to the middle of another segment in frame t+1 creating a merge, 3) the start of a segment in frame t can get linked to a middle of a segment from frame t-1 creating a split, 4) the end of segment in frame t can link to nothing in frame t+1 (disappear), 5) the start of a segment in frame t can get linked to nothing from frame t-1 (appear), 6) and the middle points of segments introduced for merging and splitting can link to nothing refusing a merge or a split. Each linking step has a cost function associated with it that has to be adjusted for a specific application. To reduce the number of impossible solutions and the computational complexity, cutoffs based on distance are introduced that do not allow linking between cells

and segments whose distance exceeds a physically possible threshold.

Shafique and Shah [25] present an algorithm for multi-frame tracking of cells. It uses a weighted graph that, to avoid high complexity, matches cells in the current frame with cells in only subsequent but not previous frames. For every new frame, the algorithm either extends the cell tracks or corrects a previous mismatch by using newly obtained information.

The second popular approach to tracking utilizes deformable models, wherein the linking step is performed by propagating the converged segmentation contours from previous frame to achieve segmentation of the current frame. This approach makes capturing cell shape change and movement easier; however, it works well only if cell displacements between frames are smaller than the cell diameter [16].

Li et al. [13] combine both tracking methods by segmenting each frame individually and using a deformable models approach that propagates cell regions detected in the first frame across all frames. The propagation is realized using active contours and energy functionals that combine a probabilistic motion term, a region competition term, and an edge term that prevent contours of different cells from merging. Next, it generates intermediate track segments and finally links the segments to establish complete cell trajectories. Combining both methods increases segmentation accuracy but proves to be very time consuming.

Padfield et al. [19] segment the spatio-temporal volume directly by extracting connected regions from a wavelet coefficient space. This results in a 3D set of segmented tubes representing cell trajectories. The tubes sometimes correspond to single cell tracks, and sometimes to a cluster of multiple cells. The clusters are then segmented into single cell tracks by using model-based constraints and a curve evolution algorithm that helps retain the relative area of cell regions across slices. The advantage of this method is that the segmentation algorithm can be automatically trained to recognize cell-like distributions based on measures from the single-track tubes of the dataset.

CHAPTER 3

METHOD

The input to the system is a sequence of time-sampled raw images acquired from in vitro live cell experiments. The output is a set of tracked and segmented cells suitable for further processing, such as morphological analysis, mitosis characterization or cell state determination through GFP reporter analysis. As fragmentation and merging errors appear to be essentially unavoidable side effects of segmentation [17], this work attempts to reduce the number of errors by utilizing the output and error detection from the tracking algorithm to re-segment the problematic cells. The overall method is presented in Fig. 3.1 and consists of an iterative process of object tracking, error detection, and error correction, until all correctable errors have been eliminated.

An existing bright field image segmentation method [1] is employed in the initial segmentation and a modified version for re-segmentation. Tracking and subsequent error detection are based on the topological alignment technique described in [20]. Errors can be of two types - over-segmentation (fragments) wherein two cells are detected instead of one, and under-segmentation (merges) wherein one cell is detected instead of two. The error correction process begins with correcting fragmented cells and uses a modified segmentation approach to correct the merged cells. Each step in the method is described in more detail below.

3.1 Bright Field Image Segmentation

Bright field microscopy images do not have the same contrast difference between the background and the cells as fluorescent microscopy images. Therefore, in-focus cells are nearly invisible. To ameliorate this problem, bright field cell detection methods utilize multiple images taken at different focal planes above and below the in-focus plane. The

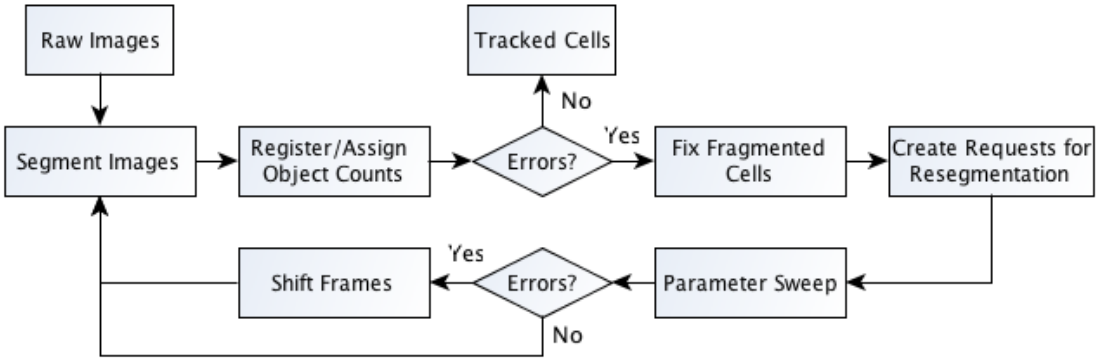

Figure 3.1. **Method overview.**

segmentation algorithm used in this work is sephaCe [1] which presents a series of algorithms for identifying and segmenting the boundaries of cells using bright field images. The first step identifies the in-focus image that serves as a reference point for the symmetrically defocused images. Since the in-focus image usually appears monotone, the image with the minimum entropy is selected. Then, two symmetrically defocused images are selected from either side of the in-focus image. The next step finds the difference of the defocused images wherein each cell is represented as an easily identifiable blurred region. The difference image is then segmented using Otsu's thresholding method [18].

To detect cell boundaries, local phase and local orientation images are generated from the bright field difference image. Local phase (shape) and local energy (magnitude) are commonly used in signal processing and are found by applying an even band-pass filter and its Hilbert transform to a signal at every point on the image. The result is a measure of the odd or the even symmetry of the signal feature. This information is used to guide the evolution of a level-set based active contour that produces cell boundary segmentations. The results from the cell detection algorithm initialize the distance function ϕ which is evolved using the level set PDE $\frac{d\phi}{dt} + F|\Delta\phi|$ where the speed term $F = \alpha F_{phase} + \beta F_{orientation} + \gamma F_{smooth} + F_{neighbor}$. F_{phase} computes the probability distribution function over the local

phase image, $F_{orientation}$ uses local orientation information to drive the level set in the correct direction, F_{smooth} is a curvature-based regularising term, and $F_{neighbor}$ is a repulsion term prevents the cells from merging.

3.2 Registering Adjacent Frames

Our cell tracking approach follows the independent segmentation and association model. The results from the segmentation algorithm described above are used to initialize the tracking system that performs frame-by-frame association of cells. Each adjacent frame pair is analyzed, and a bipartite graph linking objects in the two frames that represents likely identity relations is constructed. The bipartite graph is an undirected graph with two disjoint sets of vertices corresponding to segmented cells in respective frames and edges that connect every vertex in the first frame to a vertex in the second frame. Traditionally, in minimum flow problems, this graph contains two kinds of edges: *direct*, which maps a single object in frame t with a single object in frame $t+1$, and *split*, which maps a single object in frame t with two object in frame $t+1$ and represents the process of mitosis. This work introduces three new relations (similar to [20]): *arrive*, wherein an object is not present in frame t but is present in $t+1$; *leave*, wherein an object is present in frame t but not present in $t+1$; and *merge*, which maps two objects in frame t to one object in $t+1$ representing a potential fragmentation error in the first frame. The *split* edge is also extended to represent the case in which a single cell in frame t incorrectly splits (not mitosis) into two cells in frame $t+1$.

Initially, the system considers all possible choices of mappings between each possible object, then prunes the graph using likelihood scores and a best-first limited lookahead search [4]. Possible objects are defined as all objects that appear in a pre-specified radius around the object, wherein the radius can be easily adjusted. Likelihood scores are based on Euclidean distance and differences in shape, overlap, and movement [5]. The pruning process terminates when each object has exactly one incoming and one outgoing relation. However, this is not always possible because of segmentation and tracking errors, wherein the image has more (over-segmentation) or less (under-segmentation) objects than the actual number of cells. The next step attempts to solve the problem by assigning different counts to

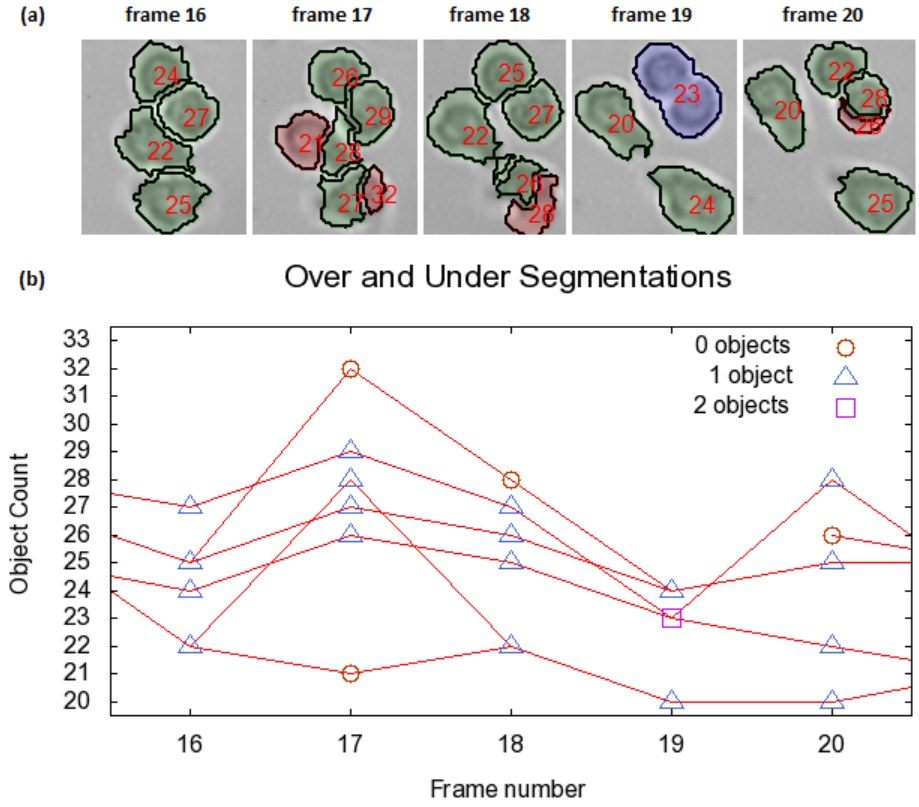

Figure 3.2. **Topological alignment:** (a) Original segmentation of a small region showing identified objects and their detected error assignment: red objects are over-segmented, blue objects are under-segmented, and green objects correctly recognized as cells. (b) The registration graph where the x-axis is the frame number and the y-axis is the object ID corresponding to the object ids in (a) above. Edges connecting adjacent frames track the objects across frames as they flow, merge or split.

objects, so each object with count one has an incoming and an outgoing edge.

An example graph is shown in Fig. 3.2 for the first iteration of the method, wherein split mappings are determined to represent over-segmentation errors rather than mitosis; and merge mappings represent under-segmentation errors. Fig. 3.3 shows the registration graph after the final iteration, in which initial segmentation errors have been detected and corrected, and all cells have been correctly tracked.

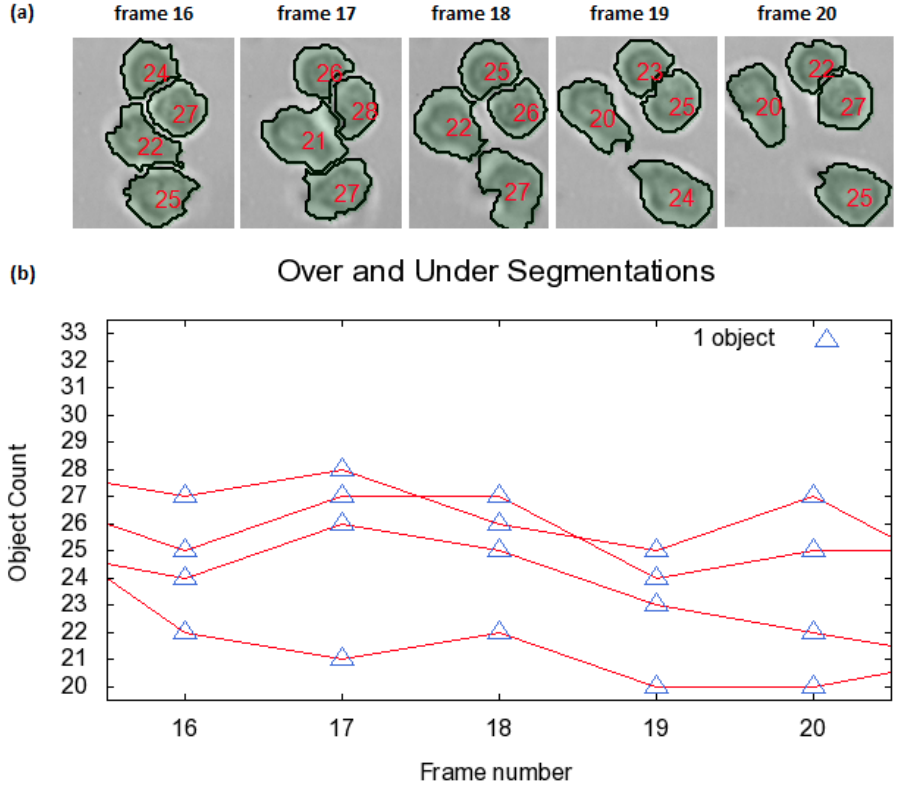

Figure 3.3. **Corrected topological alignment:** (a) The corrected segmentation and tracking of the same small region illustrated in Fig. 3.2. The fragments in frames 17, 18, and 20 have been merged with their matched cells to form continuous cells, and the under-segmented object 23 in frame 20 has been re-segmented and split into two cells. (b) The registration graph that correctly tracks identified cells across the frames.

3.3 Assigning Object Counts

The next phase of the method is a constraint optimization that uses the constraints present in the registration graph to determine how many cells each segmented object contains. A constraint optimization problem is defined by a set of variables, $X_1, X_2, ..., X_n$, and a set of constraints, $C_1, C_2, ..., C_m$. Each variable has a nonempty domain D_i of possible values, and each constraint specifies valid relations among values for some subset of variables. A solution is an assignment to all variables that satisfies all the constraints and maximizes or minimizes an objective function [23].

Let σ^t be an object in frame t, and let $c(\sigma)$ be the count of cells contained within σ. The set of variables in our problem is the set of objects in the frame. For each relation between the objects across adjacent frames (direct, arrive, leave, split, merge) the following constraints can be asserted:

$$\begin{aligned}
\mathit{direct}(\sigma_i^t, \sigma_j^{t+1}) &\Rightarrow c(\sigma_i) = c(\sigma_j) \\
\mathit{split}(\sigma_i^t, \Omega) &\Rightarrow c(\sigma_i) = \textstyle\sum_{\sigma_i \in \Omega} c(\sigma_i) + m(\sigma_i) \\
\mathit{merge}(\Omega, \sigma_k^{t+1}) &\Rightarrow c(\sigma_k) = \textstyle\sum_{\sigma_i \in \Omega} c(\sigma_i) \\
\mathit{arrive}(\sigma_i^{t+1}) &\Rightarrow c(\sigma_i) \in \{1, 2, 3\} \\
\mathit{leave}(\sigma_i^t) &\Rightarrow c(\sigma_i) \in \{1, 2, 3\}
\end{aligned}$$

where $m(\sigma_i) = 1$ if this split is mitosis, 0 otherwise, and Ω is a set of objects in a frame where $2 \leq |\Omega| \leq 3$.

For any registration graph, the produced set of equations is under-constrained since it defines only the relationship among object counts but not individual counts. An ambiguity arises, for example, when the split constraint is applied and can mean that either mitosis or over-segmentation is occurring. The merge constraint can also be ambiguous, signifying either a merge between two cells or a merge between an over-segmented object and a cell. To correct this, scores are assigned to each object giving it a likelihood of containing 0 through 3 cells, thereby turning the count assignment problem into a constrained optimization problem. The scores are based on a match between the morphology of the object and the morphology of known cells, such as the relationship between the area and perimeter of the object. Standard morphologies that are specific to cell types can be used and provide a means to incorporate prior knowledge into the system. The split ambiguity problem can then be solved by considering either one of the constraints ($m(\sigma_i) = 1$ or $m(\sigma_i) = 0$), thus either exploiting the scoring function of the cells involved or bounding the number of mitosis events allowed over the time periods considered.

Various methods can be used to solve constrained optimization problems including linear programming [6, 12] or combinatorial optimization combined with constraint propa-

gation and search effort bounding. In this work, the latter approach is used because of its flexibility and reasonable efficiency compared with linear programming. Our method uses a greedy approach by assigning a count to the object with the best score and propagating the assignment to previous and next frames, while ensuring that all flow constraints are imposed. If a better assignment is found later on, a backtracking search is applied to select the better count assignment.

The results of this analysis are illustrated on the registration graph and corresponding cropped image frames that show each object's cell count in Fig. 3.2 and Fig. 3.3.

3.4 Identifying and Correcting Errors

After the object count assignment phase terminates, every object has an assigned count. Correctly segmented cells are assigned a count of $c(\sigma) = 1$, under-segmented objects a count $c(\sigma) > 1$, and over-segmented objects a count $c(\sigma) = 0$. Once errors have been identified, correction methods are applied to the segmented frames through local replacement of segmented objects. Over-segmentation errors are corrected as part of the tracking system, while under-segmentation errors are corrected during the re-segmentation process.

3.5 Over-Segmentation Error Correction

Segmented objects that are assigned an object count of zero are fragments that must either be merged or eliminated. To correct these over-segmentation errors, it is necessary to identify an object that splits or merges into the fragment. Since the bipartite graph edges connect all matched objects, we can recursively search the graph edges starting from the fragment backward in time until we find an object that splits into two, or forward in time until we find an object that merges with the fragment. If none is found, the fragment is eliminated; otherwise, we follow the edge from the split or the merge found in the last step, until we are back on the frame with the fragment. The object on that frame, found by following the edge from the split or the merge, is the second half of the fragment. The fragment is then merged with that object (see Fig. 3.4) using a flood fill algorithm within the containing convex hull surrounding both cells.

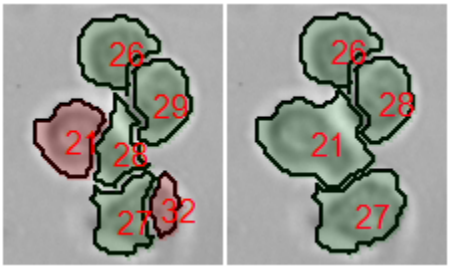

Figure 3.4. **Merging objects:** The red cells on the left represent over-segmented objects that need to be merged. The image on the right shows the result after the flood fill algorithm has been applied and the cells have been merged with the second part of the fragment.

3.6 Under-Segmentation Error Correction

Under-segmentation errors arise because bright field segmentation methods are very sensitive to the choice of the in-focus frame and the settings of controlling parameters. When the initial segmentation is performed, each object is segmented using the same in-focus frame and parameter values. This ignores a natural heterogeneity in the height and internal structure of cells. This means that both the in-focus z-frame and the appropriate parameter values for accurate segmentation may actually be distinct for each cell. The problem is compounded by changing cell morphologies between frames.

We have developed re-segmentation techniques that allow us to correct these errors. Since the error correction method does not affect correct segmentations, we only apply the re-segmentation technique to erroneous segmentations to avoid duplication of work. To correct each detected error, we customize the focus frames and parameter values by searching until the correct segmentation is found, as defined by the target object count. These techniques are described in more detail below.

3.6.1 Isolating Objects for Re-segmentation

For each object that needs to be re-segmented, we create a binary image called an error mask. The mask has a black background consisting of 0's and one object to be re-segmented consisting of 1's. The number of masks a frame can have equals the number of over-segmentation errors contained.

Figure 3.5. **Parameter sweep:** The image on the left shows the object that needs to be split. In sephaCe, a sweep over parameter R (the radius of the morphological structuring element) is performed until the object splits into two. If the radius is increased continually, the object eventually disappears (see final frame).

3.6.2 Parameter Sweep

Most image recognition algorithms often utilize mathematical morphology operations, specifically erosion and dilation. These are shape oriented operations that as defined by [9], simplify image data, preserving their essential shape characteristics and eliminating irrelevancies. Erosion removes the boundaries of regions of background pixels making holes within the background bigger. Dilation, on the other hand, gradually enlarges the boundaries of the object. Erosion followed by dilation is known as morphological opening, which removes thin islands of object pixels smaller than the structuring element.

To split the under-segmented object into two, we also use these operations. First, we isolate the object to be corrected by applying an error mask to the image. As mentioned above, segmentation is performed using Otsu's thresholding method [18] which computes a global threshold level converting the image into an intensity image. The resulting intensity image is then converted to binary, and the binary image is eroded and dilated by the same structuring element using mathematical morphology operations. To split the merged object into two objects, we apply an increasingly larger parameter that controls the radius of the morphological structuring element in sephaCe until the object splits, as illustrated in Fig. 3.5. If the split is successful, the incorrect object is removed from the original segmentation image and the corrected split cells are added to the image instead.

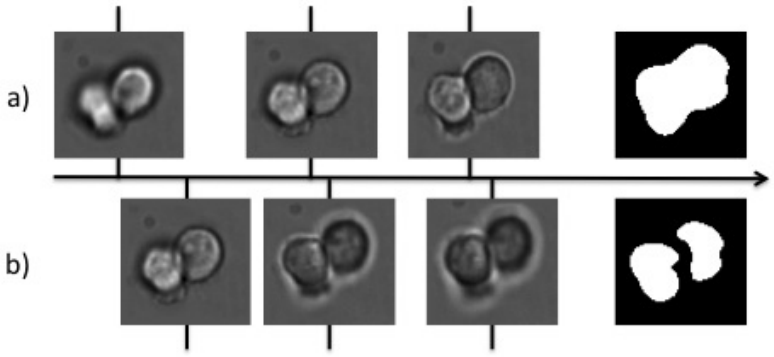

Figure 3.6. **Frame shift:** Top row shows the negative focus, in-focus, and positive focus frames respectively along the Z-axis, followed by the incorrect segmentation result. The bottom row shows the shifted negative focus, in-focus, and positive focus frames along the Z-axis, followed by the correct segmentation result.

3.6.3 Frame Shift

When the parameter sweep correction algorithm cannot detect two cells within a single object, the reason might be a difference in cell morphology, positioning or partial cell overlap, wherein one cell lies above the other (as shown in Fig. 3.6). In these cases, the choice of the global focus frame can be incorrect locally. To continue the search for a solution, alternative frames are extracted from the z-stack that are above or below the five frames used for original segmentation. Those extra frames are processed through the original segmentation algorithm and the parameter sweep method. If the correction is found, it is applied to the original segmentation by removing the incorrect object using the error mask and adding the correction to the image instead.

3.7 Feedback and Tracking Initialization

Just as topological alignment provides constraints on object counts to generate re-segmentation goals, the re-segmentation algorithms can fail to find an under-segmentation solution, because the algorithms can find no evidence that there is more than one object in that region of the image. In this case, the correction request is ignored, and the object count of one ($c(\sigma_i) = 1$) is passed back to the linking algorithm as a constraint. These kinds of errors are referred to as false under-segmentation errors; examples are shown in

Figure 3.7. **False under-segmentation parameter shift:** As in the parameter sweep in Fig. 3.5, the image on the left shows the object that needs to be split and a sweep over parameter R if performed to split the object. However, no evidence of two objects is found in the region, and the object eventually disappears without splitting.

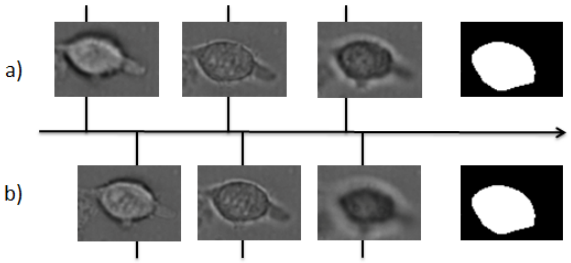

Figure 3.8. **False under-segmentation frame shift:** Top row shows the negative focus, in-focus, and positive focus frames respectively along the Z-axis, followed by the incorrect segmentation result. The bottom row shows the shifted negative focus, in-focus, and positive focus frames along the Z-axis, followed by the object that still could not be split.

Fig. 3.7 and Fig. 3.8. Discovered object counts are used to initialize subsequent iterations of tracking.

CHAPTER 4

IMPLEMENTATION

Our setup consists of two separate software tools: the first system performs segmentation of images, the second system performs tracking of cells. The segmentation system is implemented in Matlab and is an extension of the freely available sephaCe described in [1]. The tracking system is implemented in C#. We also use a set of Matlab scripts and C# programs to obtain images in necessary formats and to automate the interaction between systems. The following is a description of inputs and outputs of both systems and the interaction between them. An overview of the interaction between systems is illustrated in Fig. 4.1.

The experimental data is in the form of z-stacks, wherein each frame is photographed 30 times at different focal planes above and below the in-focus plane. The segmentation system requires five images for each frame: one in-focus, two above focus, and two below focus frames. So, the first step uses a Matlab script to extract five images at different z-dimensions from each z-stack and place them in separate folders corresponding to the image number. Next, the segmentation system is started. It extracts those images from appropriate folders and produces one output for each frame. Since this is the first iteration, no errors will be identified by the tracking system. If the segmentation system does not find any re-segmentation requests in the appropriate folder, it runs the original segmentation algorithm, skipping the error correction part. The output is converted into a grayscale image with 0's for the background and a unique integer for each object, as illustrated in Fig. 4.2.

Meanwhile, the C# folder watching program is waiting for available segmentations to start the tracking system. The tracker processes 30 frames at a time, so as the first 30 segmentation outputs become available, they are passed to the tracking system. If there

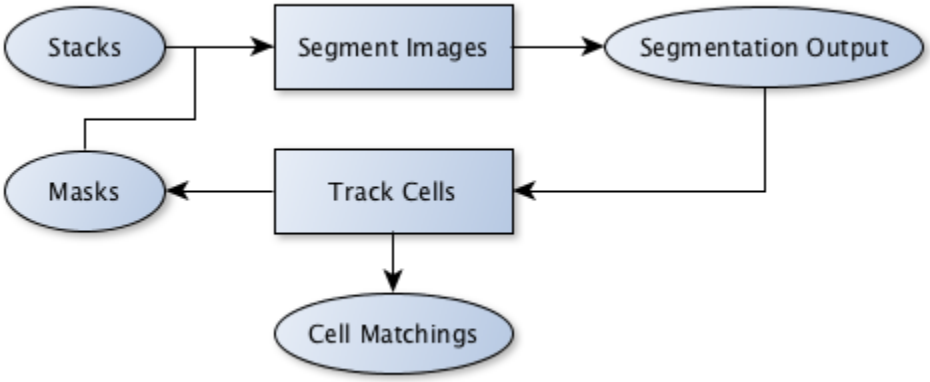

Figure 4.1. **System interaction overview:** This flowchart shows a high-level overview between segmentation and tracking systems. Circles represent inputs and outputs, and rectangles represent the systems.

are any over-segmentation errors found, the tracker corrects those errors and starts over to create new mappings between corrected objects and terminates when there are no over-segmentation errors left. Under-segmentation errors, however, need to be corrected by the segmentation system, so requests are created for each error in the form of binary masks (as described in Section 3.6.1). These are written out along with the found trajectories of cells.

As new error correction requests become available, the segmentation system is run again, but this time the original algorithm is extended with re-segmentation methods described in Sections 3.6.2 and 3.6.3. This outputs new segmentation files with corrections and feedback object counts that are fed into the tracking system, as described in Section 3.7. This continues until there are no errors remaining or can be corrected.

All computational studies were run on a 2.7 GHz Intel Core i7 processor. The original segmentation took 30 sec/frame, tracking took 5 sec/frame, and error correction took approximately 20 sec/frame.

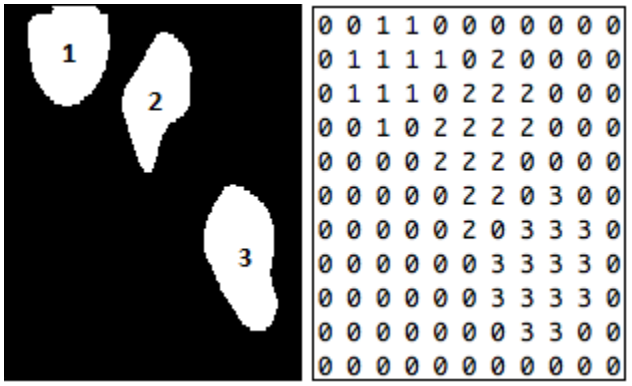

Figure 4.2. **Segmentation output example:** The left image shows an example of the segmentation output with three objects. The right image shows how the segmentation output is represented on the bitmap wherein individual cells are contiguous regions of pixels. Each cell is assigned a unique ID.

CHAPTER 5

VALIDATION OF RESULTS

To quantitatively evaluate cell and error detection and tracking accuracy, we needed a system that can capture manual ground truth annotation. Ground truth refers to the data collected from the images and represents the actual information on the image that can be compared to the automatically collected information in order to verify the system's accuracy. After trying out some open-source systems for video and image annotation, we could not find any that supported our requirements. So, in order to have all the required features, we decided to develop our own annotation system. The system consists of two parts - annotating segmentation ground truth and annotating pairwise frame tracking. The ground truth is first collected for images in both datasets. It is then compared to original segmentation and tracking results, as well as to results after every error correction iteration. Such practice shows how many errors occur when no corrections have been applied, and how the number of errors diminishes because of the correction techniques.

5.1 Validating Segmentation

To validate segmentation, the centroid positions of all visible cells on images in each dataset were manually identified using an interactive system we developed (see Fig. 5.1). The human eye can better identify overlapping cells and distinguish cells from background and other objects (e.g., glass scraps, air bubbles, etc.) that may exist in the field. After the frame is annotated, an image containing cell centers is written out as a binary mask with 0's as background and 1's as cell centers. The centers on every frame represent the ground truth segmentation that is compared to the automated segmentation results. The segmentation results are compared with the ground truth masks after every repair iteration. For every cell center coordinate in the ground truth, the segmentation output is searched at the same

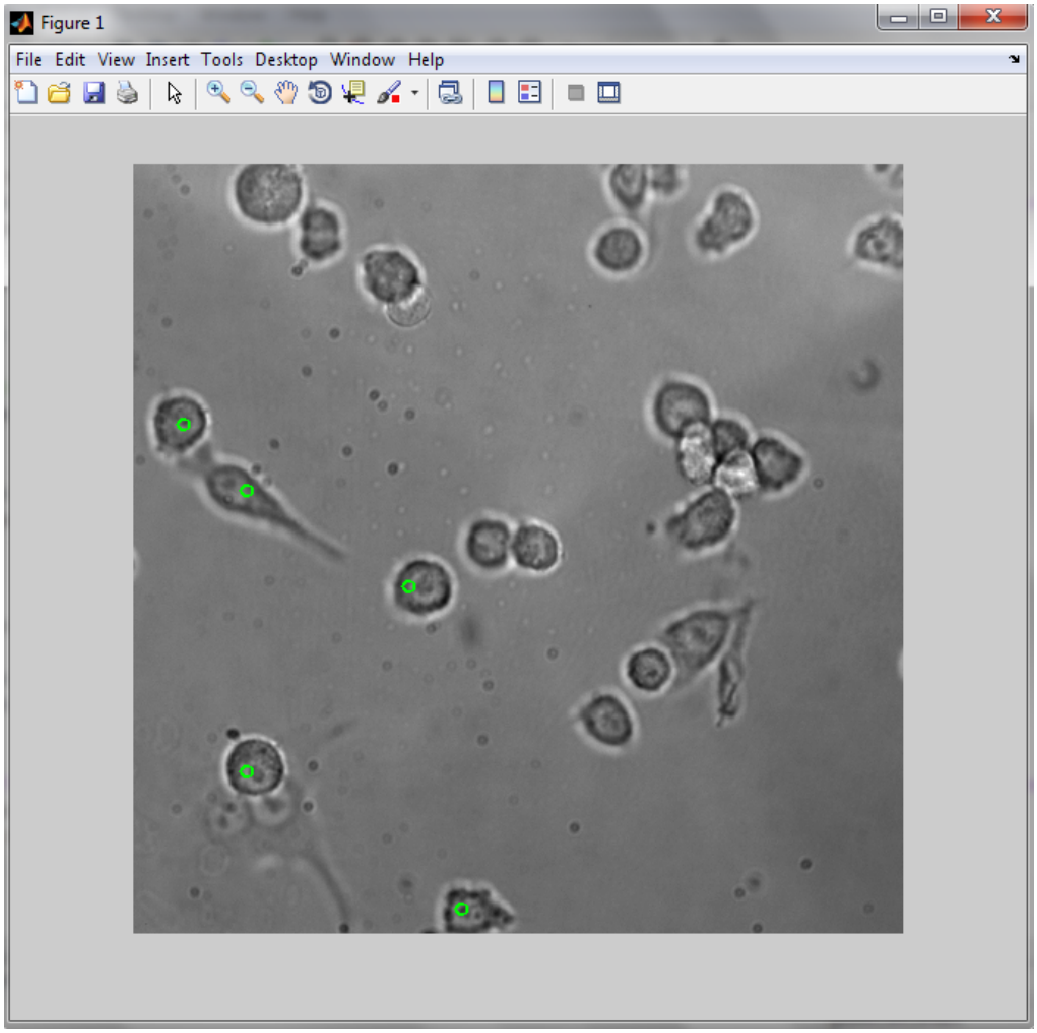

Figure 5.1. **Segmentation validation system screenshot:** The system shows an image that needs to be annotated. The green circles represent identified cells.

location for an existence of a cell. There are four possible outcomes, as summarized in Table 5.1.

5.2 Validating Tracking

To validate segmentation, we developed a system that allows the annotator to record mappings (direct, arrive, leave, split) between cells in every pair of frames. First, the initial frame is annotated with boxes around each cell by clicking on the cells. Next, the

Table 5.1. Segmentation Validation Results.

Result Type	**Formula**	**Description**
Correct Segmentation (True Positive)	$\exists\ center \in object$	Object contains exactly one cell center
Missing Segmentation (True Negative)	$\exists\ center \notin object$	Object does not exist around a cell center
Under Segmentation (True Negative)	$\exists\ center_1 \wedge \exists\ center_2 \in object$	Object contains more than one cell center
Over Segmentation (False Positive)	$\nexists\ center \in object$	Object does not contain a cell center

annotated frame is displayed on the left, while the next frame is displayed on the right. The annotated cells on the left frame are matched with new cells on the right frame, and relationships between cells are recorded using coordinates for each cell. The process is continued until all pairs of frames have been matched.

The next step after ground truth tracking has been collected for all cells, is to compare the ground truth results to results from our automatic tracking system. For each mapping in ground truth, we check if the tracker has found the same mapping between the same cells. Validation mappings use coordinates, while tracking mappings use cell IDs. So, in order to compare mappings, we need to obtain the coordinates corresponding to cell IDs. Because the validation is done on the same images as tracking, the coordinates are equal for both. That means for each validation mappings coordinates, we can access the pixel at those coordinates in the segmentation output to check if the coordinates fall within object bounds. That allows us to match ground truth cells to segmented objects. The correspondence between the coordinates and objects is recorded to avoid duplication of work. Then, the mapping between cells is compared to the mapping between corresponding objects. Every mapping in the tracking output that does not have an equivalent mapping in ground truth is considered an error. Two types of errors are possible: segmentation errors, wherein an object corresponding to some cell coordinates does not exist, and tracking errors, wherein cells are recognized correctly but the relationship between them is not the same as

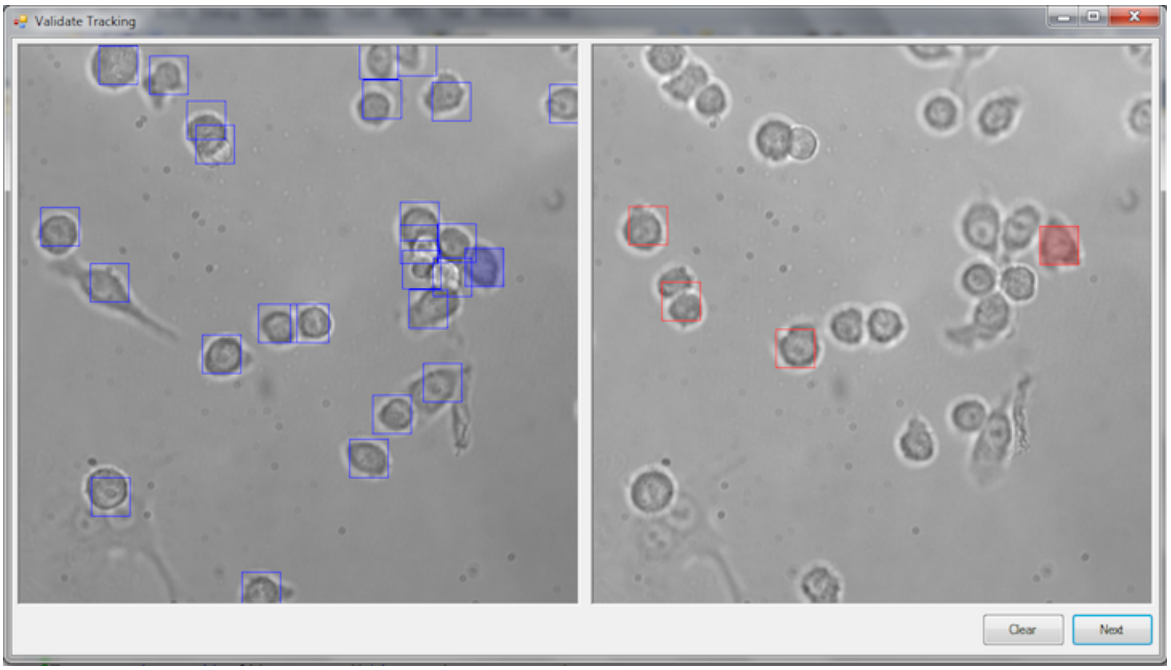

Figure 5.2. **Tracking validation system screenshot:** The blue boxes on the left image show cells that need to be matched. The red boxes on the right image show cells that have been matched. The color-filled boxes show the cells that are being matched.

the ground truth relationship.

Since the percentage of corrected errors is directly correlated with how many errors are correctly identified and reported, we also measure the correctness of the error detection system. This is measured as the number of errors that have been identified compared to the actual number of errors in the segmentation and tracking output.

CHAPTER 6

RESULTS

We performed an experimental study using two series of time-lapse images from different cell types. The first data set is HMLER human breast cancer cells sampled with 324 images, taken at 5-minute intervals. The second dataset is human embryonic kidney cells (HEK 293T) sampled with 193 images, taken at 5-minute intervals. We evaluate the effectiveness of our method by counting the number of errors before and after corrections are applied. The evaluation is performed by comparing results from segmentation and tracking algorithms to the ground truth obtained beforehand (as described in Chapter 5). There are two stages in the process - evaluating segmentation and evaluating tracking. However, these two stages are interconnected: corrected segmentation results depend on the computed errors, which are identified in the tracking algorithm, and tracking results depend on segmentation results, since the tracking algorithm cannot find correct relationships between cells if those cells are not found by the segmentation algorithm. Following is a detailed description of different types of errors and methods of evaluation.

Segmentation results represent the number and shapes of cells found on each image. Hence, when evaluating the results, we want to find the number of correctly identified cells, the number of cells that were not identified (missing), and the number of cells that were incorrectly split or merged together. To calculate those numbers, we overlay the segmentation results over the cell centers identified in ground truth images. Correct identification means there is an object at the same coordinate as a cell center, missing identification means there is no object at the cell center coordinate, under-segmentation means there is one object for more than two cell centers, and over-segmentation means there is an object with no cell center (as summarized in Table 5.1).

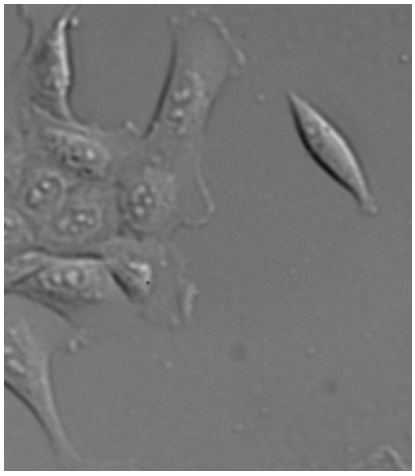

Figure 6.1. **Ground truth collection image screenshot:** This example image is used to collect ground truth. The cluster of cells on the left has low contrast, and individual cells are hard to distinguish as opposed to the cell on the right, which is very clear. Because of low contrast and the type of cells in this dataset, ground truth is not very reliable.

Figure 6.2. **Segmentation validation overlay screenshot:** The image shows the segmentation, wherein identified objects are represented as white blobs overlaid with ground truth cell centers, which are represented as red crosses. One of the cell centers does not overlap with the object causing two errors to be recorded even though two objects have been identified as expected.

There are, however, potential problems when evaluating the results using ground truth. First, ground truth is not always the absolute truth. It is manually gathered by humans, and it is not always easy to tell where the cells are. The first dataset is especially error-prone because of the type of cells that cluster together and are, consequently, hard to distinguish (see Fig. 6.1). Second, ground truth cell centers are not always exactly in the middle of the cell, and segmentation objects do not always exactly correspond to the cell shape. Even though we use an error margin of five pixels when looking for an object by looking at the cell center coordinate plus/minus 5 pixels in all four directions, there is still the potential

of the cell center not overlapping with the object it is representing (see Fig. 6.2). This is counted as two errors - one missing cell and one over-segmentation. This problem is especially relevant after the correction stage, since when under-segmented cells are split, they usually shrink in size and move away from the common center of the under-segmented object.

The graphs in Figs. 6.3-6.6 show the number of correct segmentations and the number of errors before and after corrections. The results can be misleading for the above mentioned reasons.

However, these results do not accurately represent the effectiveness of the correction methods. The number of corrections depends on the number of reported errors: not all errors are identified, and sometimes correct segmentations are incorrectly reported as errors. Incorrectly identified errors when corrected, become additional errors, thus skewing the results. Therefore, to correctly show the effectiveness of correction methods, Fig. 6.7 shows the number of identified errors and the number corrected, regardless if those are correctly identified errors or not. Some objects, however, simply cannot be split, because they are one cell in reality, as explained in Section 3.7.

The next step is to evaluate the tracking results. Tracking results represent relationships between cells in every pair of frames. The types of possible relationships are direct, split, merge, arrive and leave. The direct relationship between two cells represents the same cell in two different frames. Split can represent two cases: mitosis, when one cell splits into two cells in the next frame, and an over-segmentation error, when one cell is mapped to two objects in the next frame because the cells in the second frame are over-segmented and should be one cell in reality. Merge is the opposite of split, wherein two over-segmented cells in the first frame are mapped to one cell in the second frame. Arrive and leave relationships represent the case in which a cell enters or leaves the frame. In order to evaluate tracking results, we compare the relationships before and after segmentation corrections to the relationships gathered from the ground truth (as described in Section 5.2). Every tracking relationship found in ground truth is considered a correct result, and every relationship not

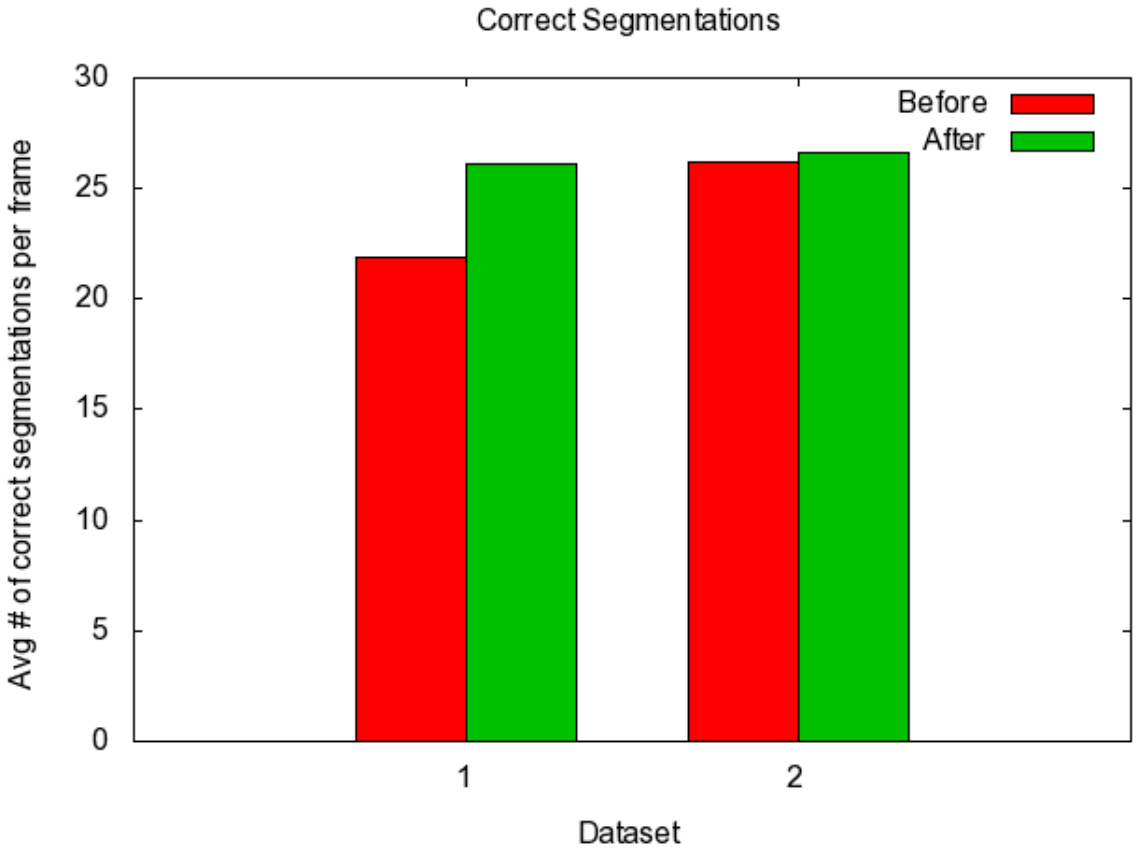

Figure 6.3. **Average number of correctly identified cells over all frames in the dataset**: The first bar shows the average number of correctly identified objects per frame before the corrections, and the second bar shows the average number of correctly identified objects per frame after the corrections.

found in ground truth is considered an incorrect result. Results are shown in Figs. 6.8 and 6.9.

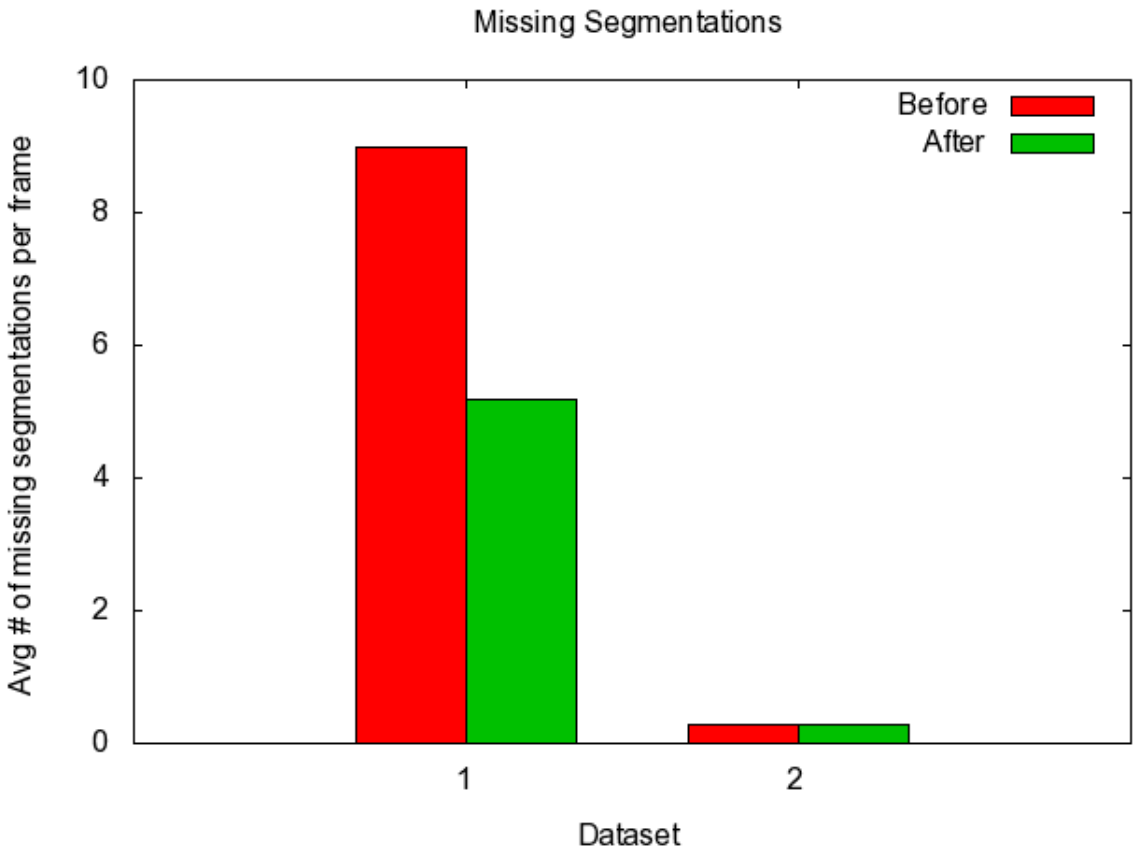

Figure 6.4. **Average number of missing cells over all frames in the dataset**: The first bar shows the average number of non-identified objects per frame before the corrections, and the second bar shows the average number of non-identified objects per frame after the corrections.

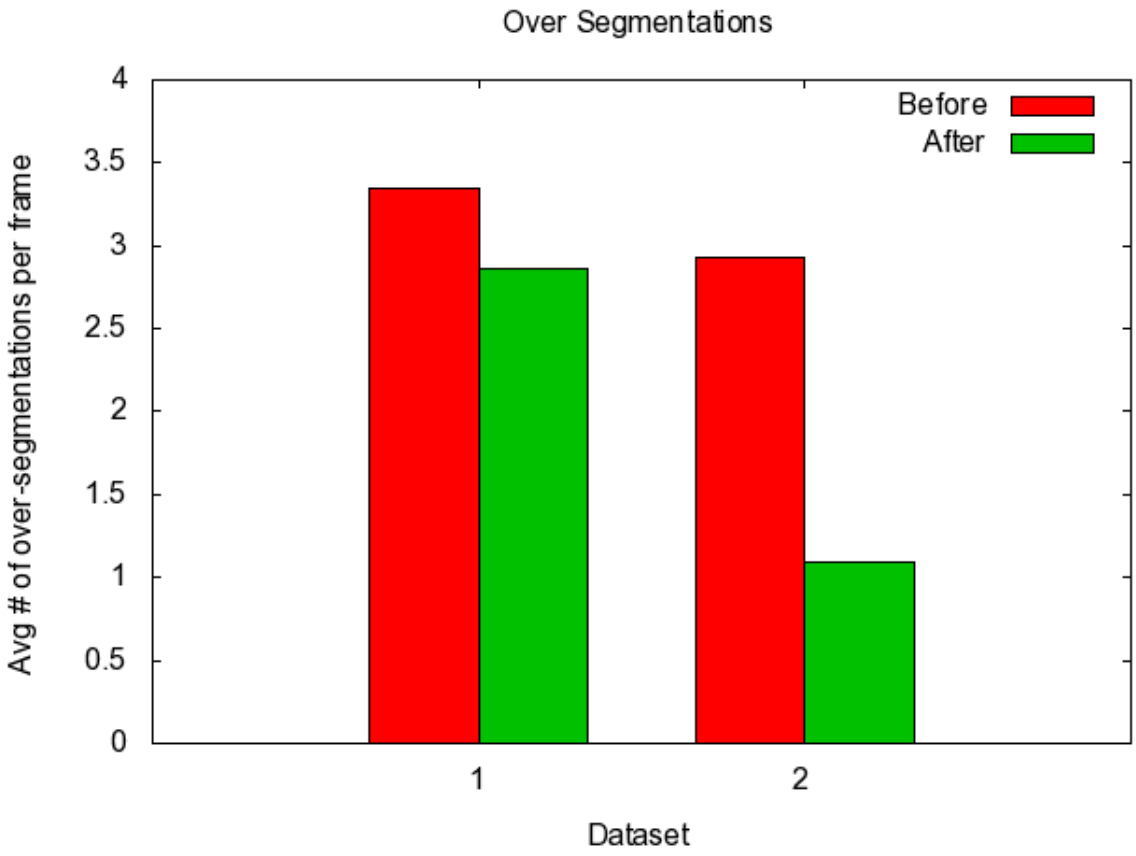

Figure 6.5. **Average number of over-segmented cells over all frames in the dataset**: The first bar shows the average number of over-segmentation errors per frame before the corrections, and the second bar shows the average number of over-segmentation errors per frame after the corrections.

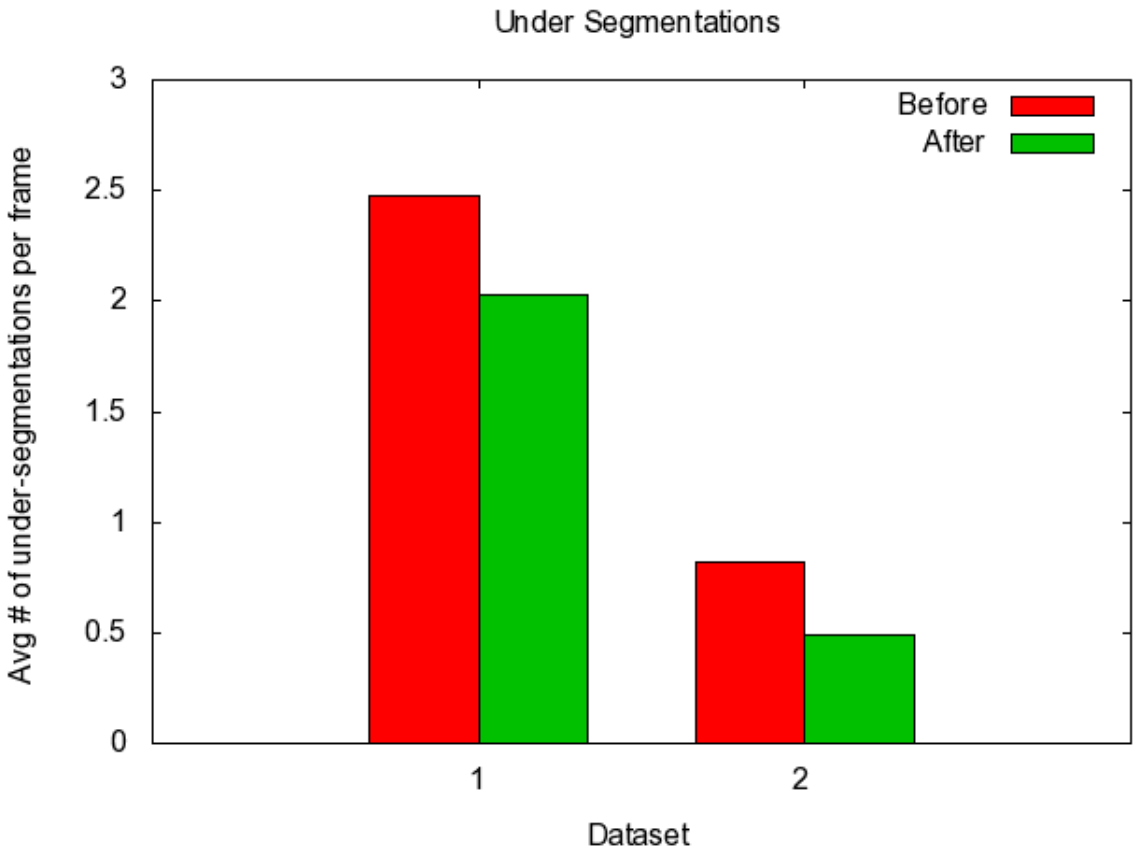

Figure 6.6. **Average number of under-segmented cells over all frames in the dataset**: The first bar shows the average number of under-segmentation errors per frame before the corrections, and the second bar shows the average number of under-segmentation errors per frame after the corrections.

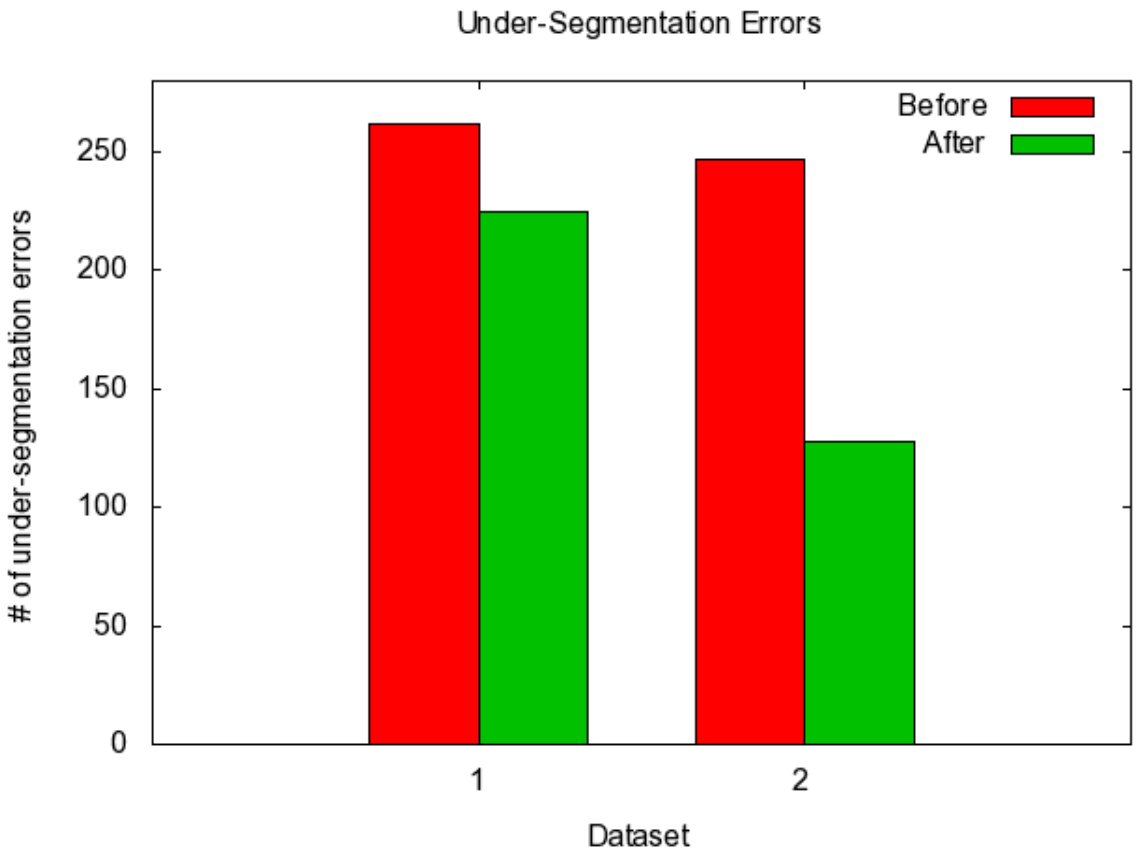

Figure 6.7. **Number of under-segmented errors**: The first bar shows the number of under-segmentation errors before the corrections, and the second bar shows the number of remaining under-segmentation errors after the corrections.

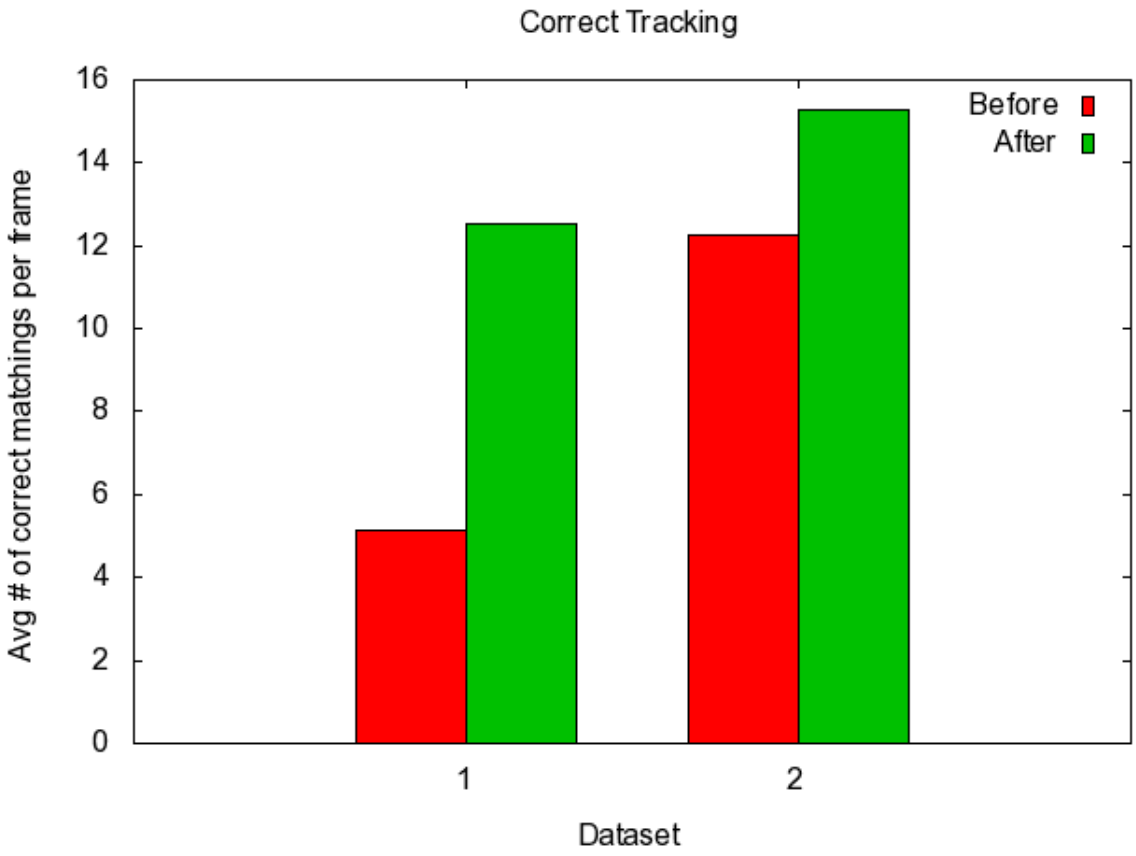

Figure 6.8. **Average number of correctly identified mappings over all frames in the dataset**: The first bar shows the average number of correctly identified mappings between objects per frame before the corrections, and the second bar shows the average number of correctly identified mappings between objects per frame after the corrections.

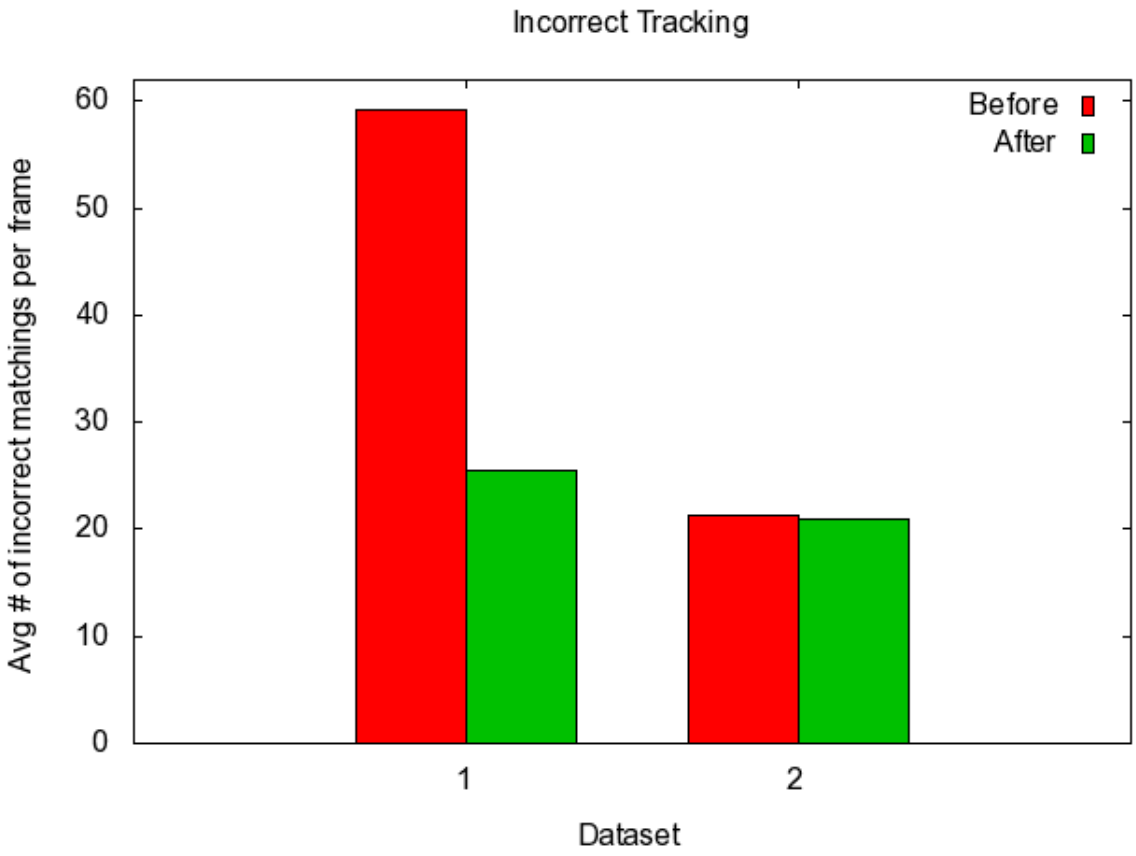

Figure 6.9. **Average number of incorrectly identified mappings over all frames in the dataset**: The first bar shows the average number of incorrectly identified mappings between objects per frame before the corrections, and the second bar shows the average number of incorrectly identified mappings between objects per frame after the corrections.

CHAPTER 7

CONCLUSIONS

A method has been presented for the correction of segmentation errors in the automated image analysis for dynamic populations of mammalian cells in time sequence images. The method is general-purpose in that it can be used for both fluorescently-tagged or bright field image sequences. It can utilize a variety of cell tracking and segmentation algorithms. A study demonstrates the method applying a topological match-based tracking approach and a recent segmentation algorithm for bright field image analysis. Rather than view segmentation and tracking as separate processes that sequentially run in a single pass, the method reported here couples both tracking and segmentation algorithms so that identified errors and their corrections form a feedback loop that runs iteratively until the residual errors are minimized.

Through error correction, sufficient numbers of tracked cells may be extracted from image data in ways that are unachievable using current processing methods. The method presented here has the potential to reduce the number of experiments required to extract information on dynamic cell behaviors, to extend the range of cell types to those that are not fluorescently-labeled, and to allow study of primary tissue culture cells, even those taken from patients.

REFERENCES

[1] Ali, R., Gooding, M., Szilágyi, T., Vojnovic, B., Christlieb, M., and Brady, M. Automatic segmentation of adherent biological cell boundaries and nuclei from brightfield microscopy images. *Machine Vision and Applications* (May 2011), 1–15.

[2] Boucaut, J., Darribre, T., Poole, T., Aoyama, H., Yamada, K., and Thiery, J. Biologically active synthetic peptides as probes of embryonic development: a competitive peptide inhibitor of fibronectin function inhibits gastrulation in amphibian embryos and neural crest cell migration in avian embryos. *Journal of Cell Biology 99*, 5 (1984), 1822–1830.

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

[4] Chen, X., Zhou, X., and Wong, S. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. *IEEE Transactions on Biomedical Engineering 53*, 4 (2006), 762–766.

[5] Comaniciu, D., Ramesh, V., and Meer, P. Real-time tracking of non-rigid objects using mean shift. *IEEE Conference on Computer Vision and Pattern Recognition*, 2 (2000), 142–149.

[6] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, 2nd ed.* The MIT Press, 2001.

[7] Dufour, A., Shinin, V., Shahragim, T., Guillen-Aghion, N., Olivo-Matin, J., and Zimmer, C. Segmenting and tracking fluorescent cells in dynamic 3d microscopy with

coupled active surfaces. *IEEE Transactions on Image Processing 14*, 9 (2005), 1396–1410.

[8] Forster, R., Schubel, A., Breitfeld, D., Kremmer, E., Renner-Muller, I., Wolf, E., and Lipp, M. Ccr7 coordinates the primary immune response by establishing functional microenvironments in secondary lymphoid organs. *Cell 99*, 1 (2000), 23–33.

[9] Haralick, R. M., Sternberg, S. R., and Zhuang, X. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9* (Apr. 1987), 532–550.

[10] Jaqaman, K., Loerke, D., Mettlen, M., Kuwata, H., Grinstein, S., Schmid, S. L., and Danuser, G. Robust single-particle tracking in live-cell time-lapse sequences. *Nature Methods 5*, 8 (July 2008), 695–702.

[11] Korzynska, A., Strojny, W., Hoppe, A., Wertheim, D., and Hoser, P. Segmentation of microscopic images of living cells. *Pattern Analysis and Applications 10*, 4 (2007), 301–319.

[12] Li, F., Zhou, X., Ma, J., and Wong, S. T. Multiple nuclei tracking using integer programming for quantitative cancer cell cycle analysis. *IEEE Transactions on Medical Imaging 29*, 1 (2010), 96–105.

[13] Li, K., Miller, E., Chen, M., Kanade, R., Weiss, L., and Campbell, P. Cell population tracking and lineage construction with spatiotemporal context. *Medical Image Analysis*, 12 (2008), 546–566.

[14] Meijering, E., Dzyubarchyk, O., Smal, I., and Cappellen, W. Tracking in cell and developmental biology. *Seminars in Cell and Developmental Biology 20* (2009), 894–902.

[15] Meijering, E., Smal, I., and Danuser, G. Tracking in molecular bioimaging. *Signal Processing Magazine, IEEE 23*, 3 (May 2006), 46–53.

[16] Meijering, E., Smal, I., Dzyubarchyk, O., and Olivo-Marin, J. *Time-Lapse Imaging.* Elsevier Academic Press, 2008, pp. 401–440.

[17] Mosig, A., Jager, S., Wang, C., Nath, S., Ersoy, I., Palaniappan, K., and Chen, S. Tracking cells in life cell imaging videos using topological alignments. *Algorithms for Molecular Biology 4* (2009), 10–19.

[18] Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, And Cybernetics 9*, 1 (1979), 62–66.

[19] Padfield, D., Rittscher, J., and Roysam, B. Spatio-temporal cell segmentation and tracking for automated screening. *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (2008), 376–379.

[20] Padfield, D., Rittscher, J., and Roysam, B. Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis. *Medical Image Analysis* (2010), 374–385.

[21] Radisky, D. Epithelial-mesenchymal transition. *Journal of Cell Science 118* (2005), 4325–4326.

[22] Reid, D. B. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control 24* (1979), 843–854.

[23] Russell, S., and Norvig, P. *Artificial Intelligence A Modern Approach, 2nd ed.* Prentice Hall, 2003.

[24] Selinummi, J., Ruusuvuori, P., Podolsky, I., Ozinsky, A., Gold, E., Yli-Harja, O., Aderem, A., and Shmulevich, I. Bright Field Microscopy as an Alternative to Whole Cell Fluorescence in Automated Analysis of Macrophage Images. *PLoS ONE 4*, 10 (Oct. 2009), e7497+.

[25] Shafique, K., and Shah, M. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 1 (2005), 51–65.

[26] Vincent, L. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing 2*, 2 (1993), 176–201.

[27] Vincent, L., and Soille, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 6 (1991), 583–598.

[28] Wahlby, C., Lindblad, J., Vondrus, M., Bengtsson, E., and Bjorksten, L. Algorithms for cytoplasm segmentation of fluorescence labelled cells. *Analytical Cellular Pathology 24* (2002), 101–111.

[29] Yan, P., Zhou, X., Shah, M., and Wong, S. Automatic segmentation of high-throughput rnai fluorescent cellular images. *IEEE Transactions on Information Technology in Biomedicine 12*, 1 (2008), 109–117.

[30] Zhou, X., Liu, K., Bradley, P., Perrimon, N., and Wong, S. Towards automated cellular image segmentation for rnai genome-wide screening. *Proceedings of Medical Image Computing and Computer-Assisted Intervention* (2005), 885–892.

[31] Zimmer, C., Zhang, B., Dufour, A., Thebaud, A., Berlemont, S., Meas-Yedid, V., and Marin, J. C. O. On the digital trail of mobile cells. *Signal Processing Magazine, IEEE 23*, 3 (May 2006), 54–62.

[32] Zordan, M. D., Mill, C. P., Riese, D. J., and Leary, J. F. A high throughput, interactive imaging, bright-field wound healing assay. *Cytometry 79A*, 3 (2011), 227–232.