Automated Tracking of Dividing Nuclei in Microscopy Videos of Living Cells

Evan Tice
Advisors: Alex Barnett, Amy Gladfelter, Hany Farid
Dartmouth Computer Science Technical Report TR2009-645

June 3, 2009

Abstract

Many cell biologists perform analysis of multinucleated cell data in order to better understand the mechanisms that regulate cell division. Sbalzarini, et al., have developed methods for automatically tracking nuclei in cell data in order to aid in this time-consuming analysis. In this paper, we present an implementation of the Sbalzarini tracking algorithm, introduce a new algorithm we developed which is able to identify mitosis events, and present other software tools we have developed to aid in the automated detection of nucleus data.

1 Introduction

Biologists would like to better understand the mechanisms leading to variability in the cell division cycle. Proper regulation of cell division is required for normal cell development and homeostasis. Poorly regulated cell division is a hallmark of cancer cells and other cells of interest to the research community. It is not known how much variability in the division cycle arises from stochastic (non-deterministic) differences in the population, as opposed to systematic differences.[1, 549] Many labs, such as the Gladfelter lab at Dartmouth College, use a multinucleate fungus as a model system to understand sources of variability in the cell division cycle. In the cells, nuclei have highly variable division cycles and divide asynchronously despite their close proximity in the same cell. Since nuclei in multi-nucleated cells share the same genetic content and reside within the same cytosol, one would expect these nuclei to exhibit similar behavior. By gathering large scale quantitative data about the division timing of nuclei in multi-nucleated cells, the labs hope to build predictive mathematical models that can explain the sources of variability in the cell cycle.

To follow nuclei in living cells, one can express a protein that localizes to the DNA, Histone, linked to the Green Fluorescent Protein (GFP). One can then visualize nuclei under fluorescence microscopy. Motorized and automated fluorescence microscopes can capture time-series cross sectional images or 'movies' of the nuclei over several hours or days. Biologists then interpret the data to better understand cell behavior.

Imaging and tracking specific cell components poses several problems. Biologist must strike a delicate balance between obtaining good images and killing the cells they wish to observe. To preserve cell viability, one must take care to limit light exposure from the microscope and balance the effects of phototoxicity against various image quality problems: Without sufficient temporal

resolution in the data, nuclei can exchange places between frames. Without cross-sectional data, nuclei can appear to collide in two dimensions. Photo-bleaching and diminished cell health can also contribute to poor image quality. Of course, nuclei also move between frames, and into and out of the frame of view. Due to these issues, it can be difficult to track nuclei between frames.

It is difficult and time-consuming for humans to manually analyze time-lapse fluorescence microscopy data. However, this is the current standard practice. A typical data set consists of images from hundreds of time points, with up to ten images for each individual moment in time. Thus, a typical data set can consist of over one-thousand images. Manually tagging nuclei by hand and extracting movement and mitosis (cell division) information is an arduous process prone to errors and human bias.

Simple single-image processing strategies such as "peak-picking" allow a user to automatically identify nuclei in a series of images. In addition, tracking methods developed by Sbalzarini, et al.[2] have been developed for tracking nuclei over time. However, the Sbalzarini tracking strategies is insufficient for dealing with mitosis events where a single particle "splits" into two; these are events of interests to biologists.

We have expanded on the work of Sbalzarini and others to develop an algorithm that is mitosis aware. That is, our algorithm not only identifies mitosis events, but uses information about identified events to determine whether mitosis is likely. We have developed a mitosis cost function which takes into account the age of particles at the time of mitosis. Our approach uses this cost function in a "simulated annealing" process to identify likely mitosis events. Simulated annealing is a statistical algorithm to search for optimal configurations in a high-dimensional space, in our case, interpretations of mitosis and movement events.

Our contributions are as follows:

- Analyzing mitosis via annealing
- Implementing a linking heuristic for the Sbalzarini algorithm
- We have developed a user-friendly software environment for performing cell analysis

In section 2 we outline the image processing pipeline they are used in. Section 3 introduces the notation we use to describe particle associations, as well as the cost functions we use in our tracking algorithms to model the likelihood of various associations. Sections4 and 5 describe our implementations of the Sbalzarini tracking algorithm and our mitosis tracking algorithm, respectively.

2 Our Software and the Processing Pipeline

In this section, we describe the software we have built and how it is used in the process of acquiring, processing, and interpreting fluorescence microscopy data. Figure 1 outlines the processing pipeline.

2.1 Image Filters

Our software takes as input images collected from the fluorescence microscope. We provide a number of image processing tools, "filters" that the allow the user to manipulate the images in order to remove background noise. We use the term "filter" loosely since certain operations, e.g., cropping, are not, strictly speaking, image filters. The user uses the tools provided to adjust the image so that the nuclei of interest appear "bright" and the rest of the image appears "dark"; the peak picking

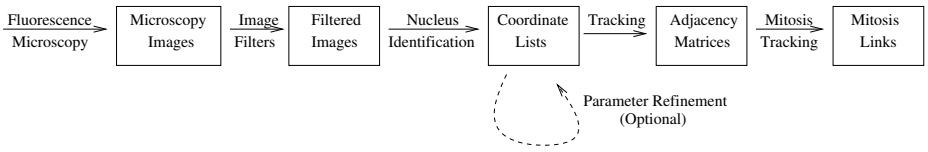


Figure 1: Processing Pipeline

algorithm described in section 2.2 assumes that the operator has processed the image in this way. The peak picking strategy works well when the operator is able to the nuclei from background noise, and when the pixels in the centers of the nuclei are brighter than the surrounding pixels. Image filters like thresholding and convolution help the operator process images with these characteristics. Some "filters" serve other purposes, e.g., the ability to crop image allows the operator to remove irrelevant pixels from the input images so system resources can be put to better use analyzing relevant portions of the input images.

Our software implements the following filters (among others):

- Cropping - Discards unnecessary pixels.
- Grayscale Filter - Convert any or all of the R,G,or B, color channels to grayscale.
- Negative Image Filter - Inverts the intensities of all pixels.
- Thresholding - Discards pixels whose intensities in the gray or, alternatively, in any or all of the R, G, or B color channels fall below a user-specified threshold.
- Binary Filter - Sets pixels to either the minimum or maximum intensity based on whether or not the pixel intensity is below or above a user-specified threshold, respectively.
- Convolution - Applies a user specified convolution kernel to the image. To convolve an image, we compute the weighted sum of pixel intensities of each pixel with respect to its neighbors and an overlay "kernel" matrix which specifies the weight of each neighbor. The UI allows the user to specify the convolution kernel matrix at runtime. In the kernel input UI, the columns of the convolution kernel are delimited by spaces the rows by newline characters. In order to obtain images where nucleus centers appear particular "bright", we often using a simple blurring convolution kernel such as that shown in Equation 1.

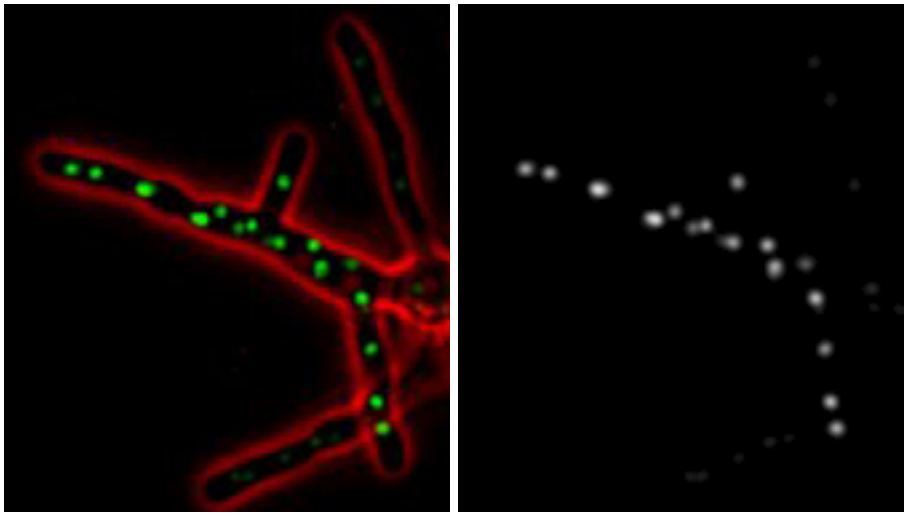
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1}$$

2.2 Nucleus Identification

We apply a "peak-picking" strategy to identify nuclei in the images. A simple algorithm analyzes each two-dimensional image and identifies pixels whose intensities are local maxima. The algorithm performs a constant amount of work for each pixel (determined by the number of neighbors considered; a user specified constant), and thus runs in time proportional to the total number of pixels

to process. The user specifies the minimum distance allowed between detected nuclei as well as a threshold intensity. We construct lists (ArrayLists) of detected nuclei coordinates for each frame. Figure 2.2 shows an example of an unprocessed image as well as the corresponding processed image.

The peak-picking algorithm only obtains parameter estimates for the position (x, y) of detected nuclei. In order to obtain size, intensity (brightness), and z-coordinate estimates, one must run an additional parameter refinement algorithm (see section 2.3).



(a) Unprocessed image (b) Processed image, ready for peak picking

Figure 2: Example image cell data, before and after image processing. The peak picking strategy works well when the operator is able to the nuclei from background noise, and when the pixels in the centers of the nuclei are brighter than the surrounding pixels.

Our nucleus identification algorithm operates on two dimensional images. In cases where multiple cross-sectional images exist for a given time-frame, the software (automatically) builds a two-dimensional composite of the the cross-section images and runs the peak-picking algorithm on this composite. In practice, this strategy tends to work fairly well with three-dimensional data, even in cases where nuclei overlap slightly. However, this heuristic does not work perfectly; our algorithm fails to detect overlapping nuclei from time to time. In section (6) we outline possible solutions to this shortcoming.

2.3 Parameter Refinement

The user may choose to apply an algorithm which refines the location estimates obtained upon the location estimates obtained by the peak-picking algorithm. This refinement algorithm can estimate size and intensity (brightness) parameters of detected nuclei. In addition, it solves for the z coordinate value when three-dimensional data are available.

The intitial parameter estimates for detected nuclei are fairly inaccurate: the location parameter is chosen according to the location detected in the two-dimensional nucleus detection algorithm.

All nuclei are initially assumed to be the same size and we assume they have the same intensities. The refinement algorithm works by building a theoretical image model of the microscopy imaging in and out of focal plane, based on the (x, y, z) coordinates, size, and intensity parameters for each nucleus.

We use a method for mode-based parameter fitting first applied to cell data by Alex Barnett [3]. We use a gaussian bump to model particle intensity in and out of the focal plane as shown in figure 3.

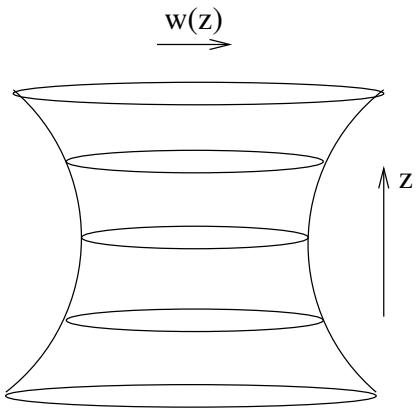


Figure 3: Gaussian bump model of focal plane. We use cross sectional data to solve for z, the particle width, and other parameters. $w(z) = sqrt(w_0^2 + s^2(z - z_0)^2)$, where s is an f-stop (speed) factor particular to microscope setup

We use the gaussian model to produce images for each frame and cross sectional depth using our parameter estimates. We compare the model images against the actual images pixel by pixel. The error for a particular set of parameter estimates is the sum of squared differences between the pixels in our actual input images and the the pixels in our theoretical model images given by our parameter estimates.

We apply a non-linear optimization algorithm[1] to iteratively optimize our parameter estimates for each nucleus in order to minimize the error. We optimize the parameter estimates for a single nucleus while holding constant the estimates for all other nuclei. We can iteratively re-apply the entire optimization process several times, since we can obtain more accurate parameter estimates for each nucleus as the estimates for overlapping nuclei become more accurate.

In minimizing the error in our model, we necessarily solve for the position (in x, y, and z), size, and intensity of each nucleus.

2.4 Tracking

Tracking enables us to preserve the identity and lineage of nuclei across frames. In the tracking steps described throughout the remainder of this paper, we describe how we attempt to match the detected particles in each frame with the related particles in the neighboring frames. We make

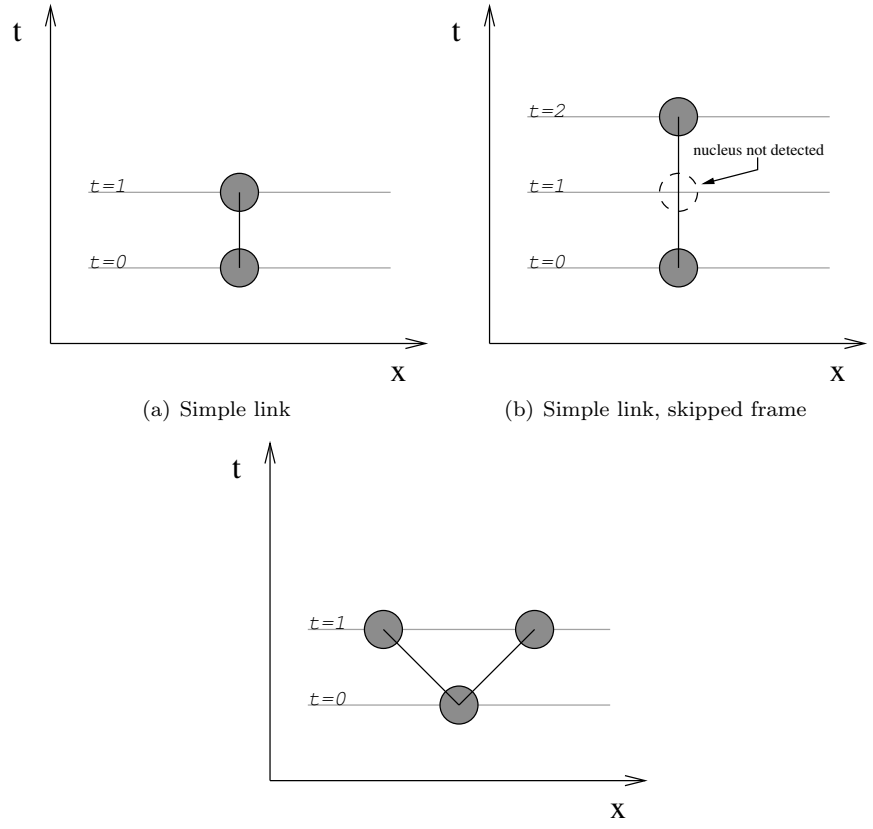
[1] We use a Nelder-Mead Simplex method implementation from a math library[4].

two types of particle associations: first, we link particles which we believe correspond to the same nuclei in different frames, and second, in the case of mitosis, we link "parent" particles to their "children.". We introduce cost functions to model the likelihood of various associations. Sections 4 and 5 describe our implementation of the Sbalzarini tracking algorithm and our mitosis tracking algorithm, respectively.

3 Notation and Cost Functions

In this section, we introduce the notation we use to describe the relationships between nuclei and the cost functions we use to model the likelihood of various interpretations of the data.

3.1 Nucleus Associations



(a) Simple link

(b) Simple link, skipped frame

(c) Mitosis link

Figure 4: Space-time diagrams denoting relationships between nuclei across frames. The y-axis represents time and each horizontal line represents a different time point. The x-axis abbreviates the (x, y) image plane on (x, y, z) in cases where three dimensional data are available.

The space-time diagrams in Figure 4 show relationships between nuclei across frames. Each light gray horizontal horizontal line in the figure represents a particular time. We depict nuclei as circles; the x position of the nuclei for a given moment in time lies at the intersection of the circle and the horizontal line for the time period. In our nucleus diagrams, a solid black line between particles in frame t and $t + r + 1$ denotes either a "simple link" (in cases involving exactly two particles) or a "mitosis" association (in cases involving exactly three particles). Cases where $r > 0$ allow us to handle nuclei which disappear for consecutive frames. The Sbalzarini implementation currently allows $r > 0$ whereas our current simulated annealing implementation does not[2].

A particle a in frame t is linked (via a "simple link") to a particle b in frame $t + r + 1$ if (and only if), in the decision produced by our software, particles a and b correspond to the same physical nucleus. A "mitosis link" between particles a in frame t and particles b, c in frames $t + r + 1$ exists if (and only if), in the decision produced by our, the physical nucleus corresponding to a divided to produce physical nuclei corresponding to b and c.

We use the notation $a \rightsquigarrow b$ to denote a link between particles a and b. Similarly, we use the notation $a \rightsquigarrow b, c$ to denote a mitosis event involving parent particle a in frame t and child particles b and c in frame $t + r + 1$.

3.1.1 "Dummy particle" (δ) associations

The Sbalzarini tracking algorithm assumes imperfection in the nucleus identification process. To account for particles which appear to appear or appear to disappear, Sbalzarini uses the notion of a "dummy" or "null" particle, denoted δ. We allow links between the dummy particle and detected particles in order to account for missed or mistakenly identified nuclei. We say that the dummy particle links to nuclei which have no predecessor in the preceding frame. The dummy particle links to all nuclei in the first frame ($t = 0$), and may link to various "appearing particles" in subsequent frames. For the purposes of this paper and our software, we do not allow mitosis associations to ever include a dummy particle.

3.2 Simple cost functions

Cost functions provide a unified framework to compare associations. We assign a cost function to each type of association. The cost functions values vary with the likelihood of various associations; they produce "low" values for likely associations and "high" values for unlikely associations. The Sbalzarini and annealing algorithms rely on cost functions to search for an optimal set of associations, the set of associations that produces the lowest total cost.

We expresses the distance between two particles a and b in terms of their coordinates (x_a, x_b, y_a, and y_b) thus: $d(a, b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$.

We consider associations between particles in frame t and frames $t + r + 1$ where $0 \leq r < r_{\max}$, where $r_{\max}$ is a user specified parameter. The user specifies the maximum allowable distance, $d_{\max,0}$ between particles in subsequent frames (i.e., when $r = 0$). In processing sample data from the lab, we find that $d_{\max,0} = 12$ pixels usually produces reasonable results. The maximum distance $d_{\max,r}$ varies by r as shown in Equation 2.

We express the cost of making an association $a \rightarrow b$ in terms of the distance function and $d_{\max,r}$ functions as shown in equation 3.

[2]Our annealing implementation could be extended to consider associations that allow skipped frames

$$d_{\max,r} = d_{\max,0} \cdot (r+1), \qquad r > 0 \tag{2}$$

$$\phi(a, b, d_{\max,r}) = \begin{cases} d_{\max,r} & \text{if } a = \delta \text{ or } b = \delta \\ \infty & \text{if } d(a,b) > d_{\max,r} \\ d(a,b) & \text{otherwise} \end{cases} \tag{3}$$

The Sbalzarini algorithm described below relies exclusively on cost function equation 3. Section 5.1 introduces defines the mitosis cost function used in the simulated annealing algorithm.

4 Sbalzarini Tracking Algorithm

The algorithm by Sbalzarini, et al., starts with a set of particle associations and iteratively optimizes this state until the resulting associations have a minimal (optimal) total cost. Our implementation, like that described by Sbalzarini, uses one adjacency matrix to store the current associations between particles in a given frame t and subsequent frame $t + r + 1$. For each frame, there are at most $r_{\max}$ association matrices. Each adjacency matrix contains one column for each particle in frame t and one row for each row in frame $t + r + 1$. An additional column and an additional row exist for the dummy particle. Each entry ij in the adjacency matrices contains a boolean value which indicates whether or not a link exists between the i^{th} in frame t and particle j^{th} in frame $t + r + 1$.

At all times, the set of associations represented by a given association matrix respects the following topology rules: "[each non-dummy row and non-dummy column in the adjacency matrix contain exactly one TRUE entry and all other entries are FALSE. The dummy rows and columns may contain more than one TRUE entry. "[2, p186]. These topology rules have the following implications:

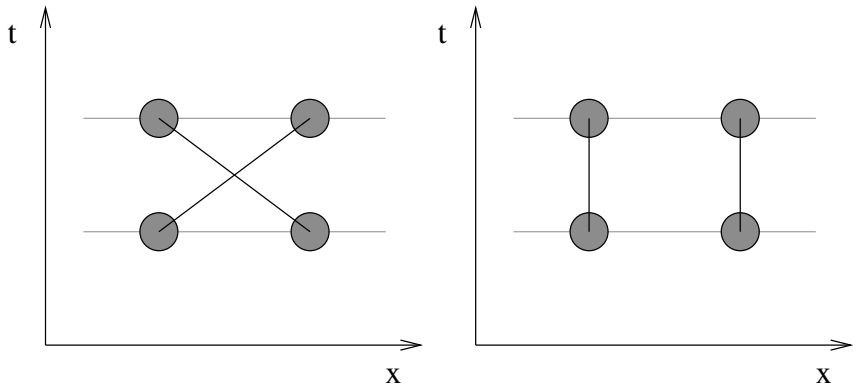
1. Each non-dummy particle can link to only one particle in subsequent frames.
2. No two particles, dummy or otherwise, can link to the same particle.
3. Multiple particles can link to the dummy particle, the dummy particle can link to to multiple particles

The initial set of associations may be chosen arbitrarily, so long as they respect the topology rules. Our implementation uses a nearest-neighbor heuristic to create an initial set of associations. Algorithm 1 contains the pseudocode for the iterative optimizations we perform the Sbalzarini tracking algorithm. The optimization step repeatedly considers swapping the particles involved in various associations as shown in figure 5, and performs the swaps when doing so results in a lower total cost. The algorithm terminates when the optimal set of associates, with respect to the cost function, is achieved.

We had difficult interpreting the strategy used by Sbalzarini to extract the set of optimal links from the adjacency matrices. We extract a set of association from the adjaceny matrices using a linking heuristic that we developed. The linking heuristic is coded in two parts: Algorithm 2 defines a 'linking loop' which calls an inner recursive step, Algorithm 3, repeatedly.

Algorithm 1 Tracking overview

repeat
  for all time frames $t = 0...T-1$ **do**
    for all adjacency matrices $r = 0...min(R, T-t-1)$ **do**
      Step 1 - Direct swaps: Consider the cost of alternative links but ignore links to/from dummy particles.
      for all nuclei i in frame t **do**
        for all nuclei j in frame $(t+r+1)$ not linked to i with $\phi(i,j,(r+1)d) < \infty$ **do**
          if $k \rightarrow j$ and $i \rightarrow l$, where j, k, i, l are all non-dummy particles. **then**
            Compute $z_{ij} = \phi(i,j,(r+1)d) - \phi(i,l,(r+1)d) - \phi(k,j,(r+1)d) + \phi(k,l,(r+1)d)$
            if $z_{ij} < 0$ **then**
              Break associations $i \rightarrow l$, $k \rightarrow j$
              Make associations $i \rightarrow j$, $k \rightarrow l$
            end if
          end if
        end for
      end for
      Step 2 - Appearing particles: As with Loop 1 above, but consider the cost of the alternative when $k \rightarrow j$ and $i \rightarrow l$, where j, k, i are all non-dummy particles but l is a dummy particle.
      for all nuclei j in frame $(t+r+1)$ **do**
        if $k \rightarrow j$ with j, k non-dummy **then**
          Compute $z_{\delta j} = \phi(\delta,j,(r+1)d) - \phi(k,j,(r+1)d) - \phi(k,\delta,(r+1)d) + \phi(k,l,(r+1)d)$
          if $z_{\delta j} < 0$ **then**
            Break associations $k \rightarrow j$
            Make associations $\delta \rightarrow j$, $k \rightarrow \delta$
          end if
        end if
      end for
      Loop 3 - Disapppearing particles: As with Loop 1 above, but consider the cost of the alternative when $k \rightarrow j$ and $i \rightarrow l$, where j, i, l are all non-dummy particles but k is a dummy particle.
      for all nuclei i in frame t **do**
        if $i \rightarrow l$ with i, l non-dummy **then**
          Compute $z_{i\delta} = \phi(i,\delta,(r+1)d) - \phi(i,l,(r+1)d) - \phi(\delta,l,(r+1)d) + \phi(k,l,(r+1)d)$
          if $z_{i\delta} < 0$ **then**
            Break associations $i \rightarrow l$
            Make associations $i \rightarrow \delta$, $\delta \rightarrow l$
          end if
        end if
      end for
    end for
  end for
  pass = pass + 1
until pass = maxNumPasses



(a) Associations with high total cost due to great distance between linked particles

(b) Optimal associations (low total cost) due to short distance between linked

Figure 5: Swaps considered by the Sbalzarini algorithm. The optimal associations are those with the lowest total cost.

Algorithm 2 Main linking loop *linkAll*

```
set visited[a] = FALSE ∀a
set links[a] = NIL ∀a
for all frames t from 0 to t_Max − 1 do
  for all nuclei n in frame t do
    if NOT visited[n] then
      recursivelyLink(n)
    end if
  end for
end for
```

Algorithm 3 Recursive step *recursivelyLink*(a)

```
Let t(a) denote the frame in which particle a exists.
set visited[a] = TRUE
set next = NIL
for all r from 0 to r_Max s.t. (t(a) + r + 1) < the total number of frames. do
  if next = NIL then
    if there exists a link between a and someb in frame (t(a) + r + 1) such that b ≠ δ and NOT
    visited[b] then
      let next = b
    end if
  end if
end for
if next ≠ NIL then
  links[a] = next
  recursivelyLink(next)
end if
```

4.1 Test results of Sbalzarini tracking algorithm with real data sets

In this section, we outline the results of the Sbalzarini tracking algorithm on several real data sets. We attempted to interpret and follow by eye the particles identified by the nucleus identification algorithm across frames, and compared our own links with those links chosen by the algorithm.

Figure 6 shows two frames of cell data with information overlays from our software. Note that pixel intensities have been inverted for visual clarity. The overlays indicate the position of detected nuclei. Our software assigns each particle a unique identifier. However, the label for each particle shown in these figures is the unique identifier assigned to the nucleus when the software believes it first appears; particles which are labeled the same in different frames are thought to be the same nucleus.

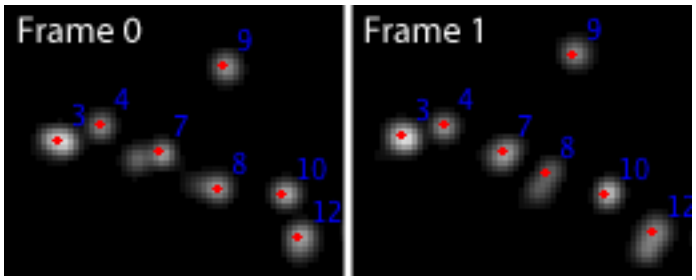


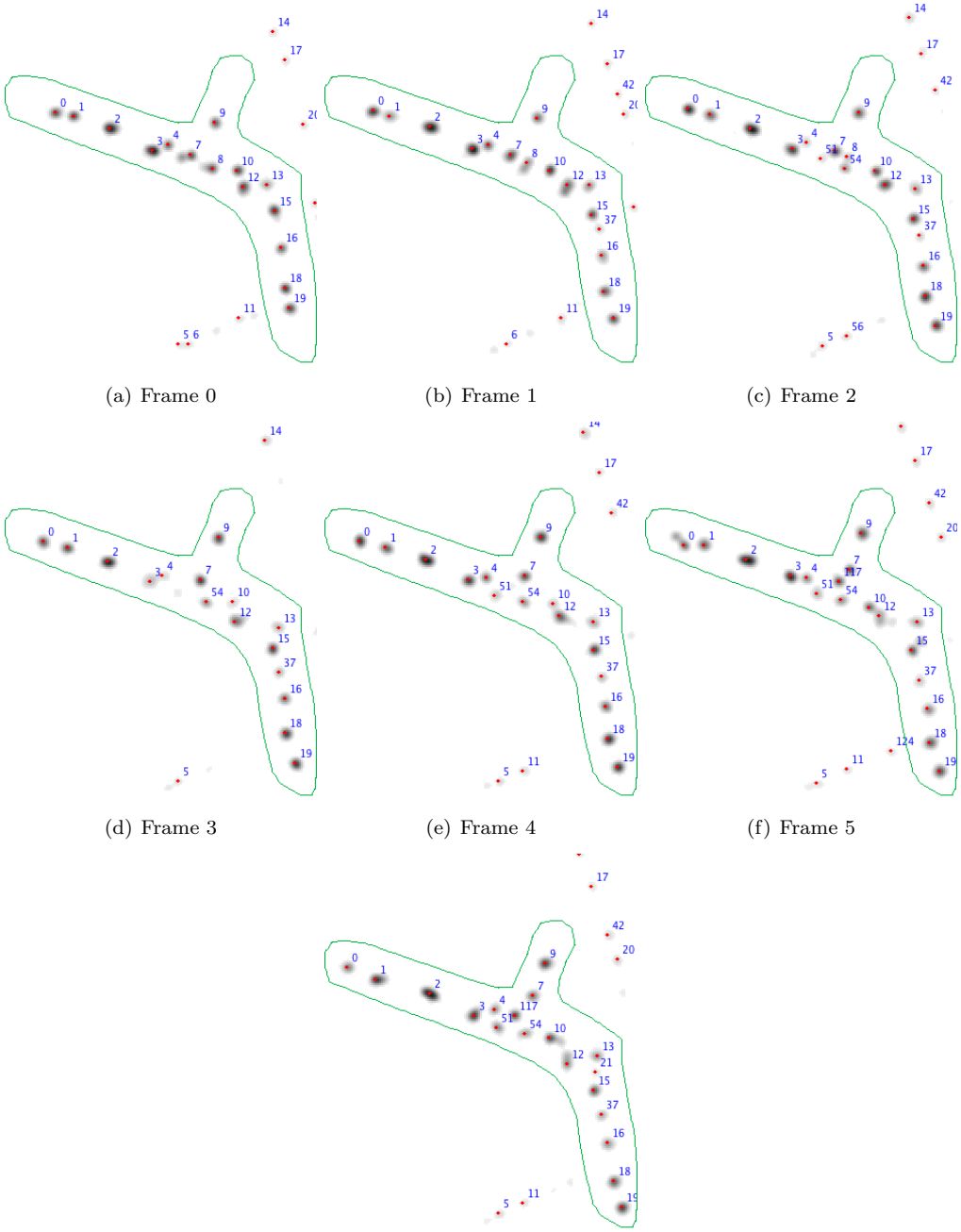
Figure 6: Tracking results produced by our software. A dot in the center of each detected nucleus shows the location of the nucleus identified by the peak picking algorithm. We particles by number in each frame; particles with the same number in different frames are thought to be the same nucleus.

4.1.1 Tracking Example 1

All data sets are from images acquired from live Ashbya gossypii cells. This filamentous fungus is expressing a histone protein, which associates with DNA, that is fused to the green fluorescent protein (GFP). The fluorescence signal is a visual reporter of the nucleus in live cells. Images were acquired on a Zeiss AxioImager-M1 upright light microscope equipped with the following Zeiss oil immersion objective Plan-Apochromat 63x/1.4NA. For visualization of GFP, Chroma filter set 41025 and Zeiss filter set 38HE were used. An Exfo X-Cite 120 lamp was employed as the fluorescent light source. Images were acquired with a Hamamatsu Orca-AG (C4742-80-12AG) CCD camera driven by either OpenLab 5 (Improvision). Z-stacks were acquired at 0.5μm slice sizes over a total of 6 microns depth at one minute intervals and processed by "fast" deconvolution using calculated point spread functions in Volocity 4 (Improvision). All still images were linearly contrast enhanced in Volocity 4. All images and movies presented are maximum projections of 3-dimensional volumes.

Figure 7 shows the output of the tracking algorithm for a seven frame data set. The images shows a cell with four hyphae. The top and bottom hyphae are out of the focal plane resulting in poor nucleus detection for nuclei in these hyphae. We only nuclei in focus, namely, the detected ones, as indicated on the figures by a green polygon[3]. We thus consider 120 of the 159 detected

[3]the green polygon denoting the figures that are in focus was added by a human interpreter, not by our software



(a) Frame 0 (b) Frame 1 (c) Frame 2

(d) Frame 3 (e) Frame 4 (f) Frame 5

(g) Frame 6

Figure 7: Sbalzarini tracking results for sample data set

nuclei. It is worth noting that the peak picking strategy failed to identify nuclei several times when nuclei were in close proximity to one another. We do not not account for these nuclei in this section since they were not considered by the tracking algorithm.

A comparison of the human and algorithm linking results for a seven frame data set follows:

1. *14 nuclei linked successfully all 7 times (i.e., 73.7% of nuclei linked correctly throughout all frames).* The nuclei labeled 0, 1, 2, 3, 4, 7, 9, 10, 12, 13, 15, 16, 18, 19 were tracked between frame 0 and frame 6 without ambiguity.

2. *1 nucleus mis-linked once, linked successfully 3 times.* Nucleus 8 begins in frame 0, disappears in frames 3 and 4, and apparently reappears in frame 5, incorrectly tagged as nucleus 117. It is correctly tracked (as nucleus 117) into frame 6.

3. *1 nucleus linked successfully 5 times.* Nucleus 37 appeared in frame 1 and is tracked without ambiguity success through frame 6.

4. *1 nucleus linked successfully 3 times.* Nucleus 51 appeared in frame 2. It appears to disappear in frame 3 (it was not picked up by the peak-picking algorithm), but reappears in frame 4 and is successfully tracked through frame 6.

5. *1 nucleus linked successfully 4 times.* Nucleus 54 appears in frame 2 and is tracked without ambiguity through frame 6.

6. *1 nucleus mis-linked once.* A mitosis child of nucleus 15 appearing in frame 6 is incorrectly tagged as nucleus 21. The correct nucleus 21 disappears (from outside the area of focus) after frame 1.

Of 19 nuclei, 14 (or 73.7%) were linked correctly throughout all frames. In total, we have $14 \cdot 7 + 3 + 5 + 3 + 4 = 113$ successful links, and 2 incorrect links. 98.26% of the links were made correctly by the Sbalzarini algorithm.

The Sbalzarini algorithm does not consider mitosis events at all. As far as the algorithm is concerned, particles which "appear" due to mitosis are no different from particles which "appear" for other reasons. In the case of the example above, four likely mitosis events were ignored by the software,

4.1.2 Summary of Sbalzarini results for all data sets

Table 1 lists the success/failure rate for the Sbalzarini algorithm against a human tracker

4.2 Analysis of Sbalzarini Tracking Algorithm

As our results show, the Sbalzarini Algorithm works well for simple tracking purposes. Sbalzarini, et al. present a detailed runtime analysis in [2, p187]. The algorithm is quite efficient since the cost function $\phi(a, b, d_{\max,r})$ does not depend upon the state of associations [2, p186]. However, this property limits the information that we can consider in the cost function. These limitations prove problematic when we wish to consider mitosis associations; in order to aid the biologists, our software needs to track division and retain lineage identity. The following section describes a tracking algorithm capable of identifying mitosis events.

Table 1: Summary of Sbalzarini results for all data sets

Data set	# Frames	Correct links (% Links)	Incorrect links (% Links)	Correct Mitosis Events (% Mitosis Events)	Missed Mitosis Events (% Mitosis Events)	Mitosis False Positives (% Links)
Sample 1	7	113 (98.26%)	2 (1.74%)	n/a	n/a	n/a
Sample 2	12	130 (100%)	0 (0%)	n/a	n/a	n/a
Sample 3	21	190 (99.48%)	1 (0.52%)	n/a	n/a	n/a

5 New Mitosis-Aware Simulated Annealing Algorithm

Simulated annealing is an optimization heuristic for solving minimization problems in high dimensional parameter space[5]. Unlike Sbalzarini, annealing is a probablistic algorithm. It tries various associations according to their cost function, and a "temperature" function, which allows high cost associations in early iterations but "cools" to favor lower cost associations over time. Due to the random element of the algorithm, associations can be made which temporarily result in high total cost, but in subsequent iterations, allow for other associations to be made or changed which result in lower total cost, as shown in figure 8. Algorithm 4 gives pseudocode for a basic annealing algorithm. In section 5.2 we present a more specific version of the algorithm tailored to mitosis detection.

Certain rules of biology provide clues as to whether or not a mitosis event logically explains a newly appearing particle. For example, nuclei are not capable of dividing immediately after their birth because they must replicate their DNA before they can undergo mitosis again. If mitosis occurs for a particular particle in a given frame, it is highly unlikely that the same particle will divide soon thereafter. Simulated annealing allows us to implement a tracking algorithm capable of accounting for properties like "time since last mitosis" when searching for an optimal set of associations. Figure 9 shows a case where a mitosis algorithm that relied only upon distance could make an incorrect decision about which nucleus gave birth to a newly appearing particle. Similarly, as shown in 10, a mitosis-aware algorithm could revisit linking decisions made by the Sbalzarini algorithm. Simple links that appear optimal with respect to the Sbalzarini cost function may be sub-optimal when considering mitosis events involving the same nuclei in past and future frames.

Our present implementation of the simulated annealing algorithm solves for the case shown in figure 9, and while it could be extended to solve for the case shown in figure 10, it does not presently do so.

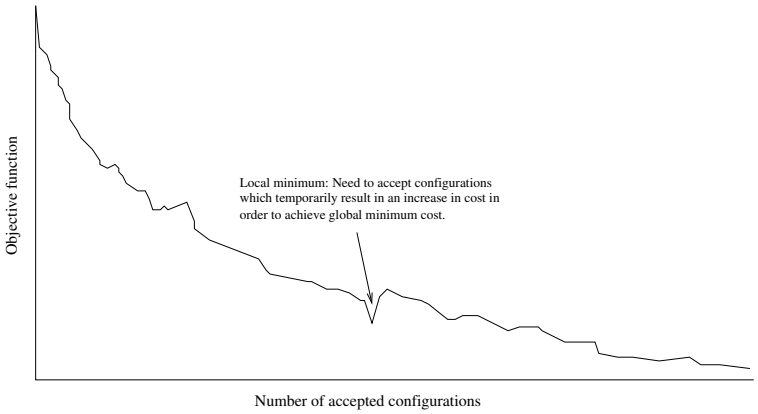


Figure 8: A "temperature" function allows high cost associations in early iterations but "cools" to favor lower cost associations over time. Due to the random element of the algorithm, associations can be made which temporarily result in high total cost, but in subsequent iterations, allow for other associations to be made or changed which result in lower total cost.

Algorithm 4 Generic Simulated Annealing Algorithm

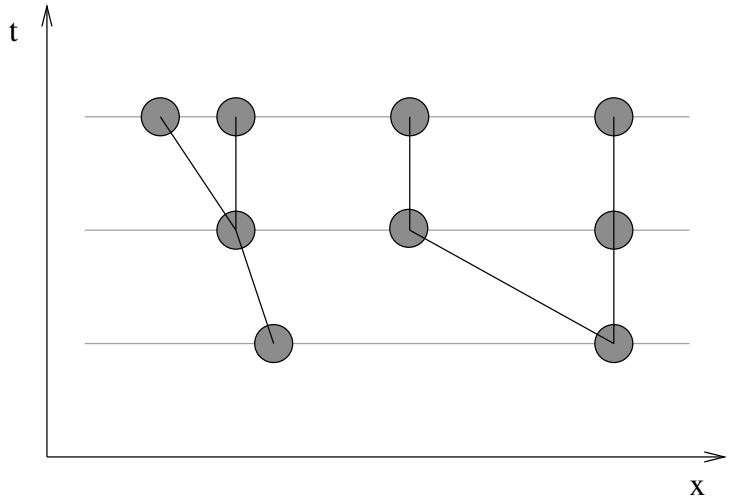
```
Let A = State Variable
Let k = 0
repeat
  Let A' = A with a small state change
  Compute change in cost Δφ between A and A'
  if Δφ < 0 or RANDOM() ≤ e^{-Δφ/TEMP(k)} then
    Set A = A'
  end if
  k = k + 1
until k ≥ k_max
```

We have developed a mitosis cost function which takes into account the age of particles at the time of mitosis. We implement a mitosis-aware algorithm which uses this cost function in a "simulated annealing" process to identify likely mitosis events and correct decisions made by the Sbalzarini algorithm that are suboptimal when considering mitosis events in past and future frames. Our algorithm takes the associations identified by Sbalzarini as input and considers introducing various mitosis associations or swaps that better explain the data.

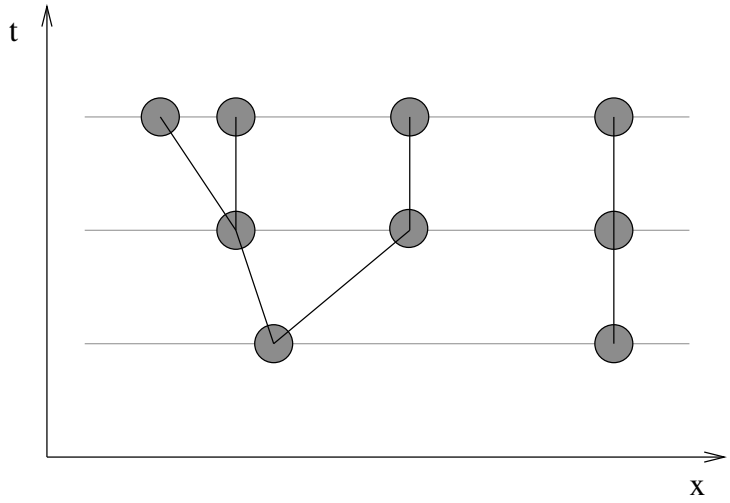
5.1 Mitosis Cost Function

In this section, we introduce a cost function which will allow us to make mitosis associations. We begin with a cost function that is not mitosis-age aware, and build upon this to construct a cost function that is aware of mitosis age.

Consider particle a in frame t and particles b and c in frame $t+1$ where $a \rightsquigarrow b$ and $\delta \rightsquigarrow c$ in

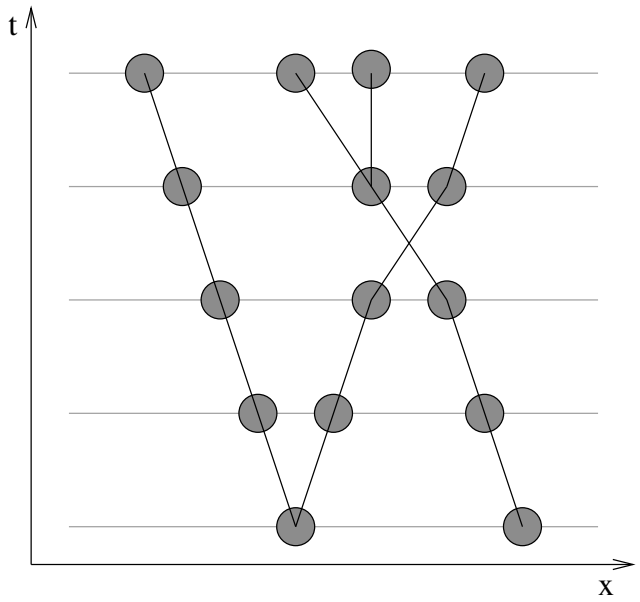


(a) Correct interpretation

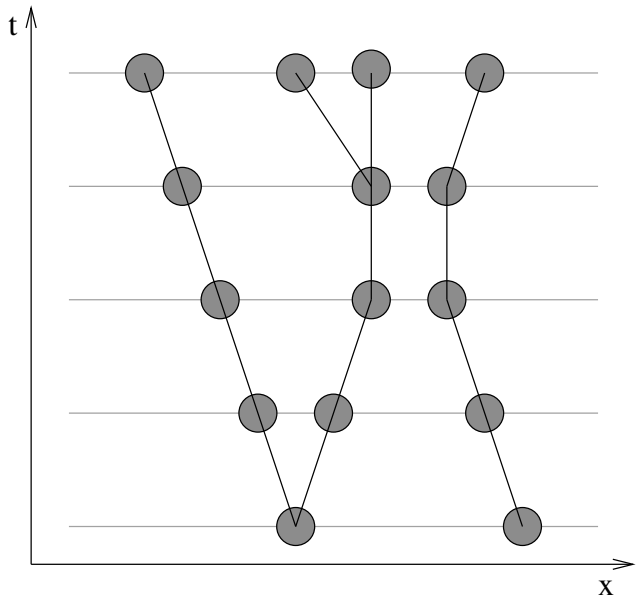


(b) Incorrect interpretation

Figure 9: Making mitosis-aware decisions: Newly appearing particle in 2$^{\text{nd}}$ frame likely came from the parent on the right that has not given birth recently (even though this parent may not be the closest match in terms of distance)



(a) Correct interpretation



(b) Incorrect interpretation

Figure 10: Resolving ambiguous "swaps": Simple links that appear optimal with respect to the Sbalzarini cost function may be sub-optimal when considering mitosis events involving the same nuclei in past and future frames

the current state.

where k_1 and k_2 are user specified mitosis cost constants expressing the fixed cost of mitosis and the distance multiplier cost of mitosis, respectively.

Let $\phi_{\max} = k_1 + k_2 \cdot d_{\max,0}$. We shall use $\phi_{\max}$ to allow us to weight the contributions of the distance and mitosis age components of our cost function relative to one another.

We define a mitosis age function $\textsc{age}(a)$ that returns the age of particle a with respect to all associations in the current state. The user provides an estimate, age_0, of the age of the particles in the first frame. In addition, the user provides a minimum age of mitosis, $\text{age}_{\min}$. The cost function includes an age penalty only when a particle younger than $\text{age}_{\min}$ divides.

Equation 4 gives the complete mitosis cost function:

$$\phi_m(a,b,c) = k_1 + k_2 \cdot \phi(a,b,d_{\max,0}) + k_2 \cdot \phi(a,c,d_{\max,0}) + \max(0, \phi_{\max} \cdot (1 - \textsc{age}(a)/\text{age}_{\min}) \quad (4)$$

where k_1 and k_2 are user specified mitosis cost constants as specified above, and where k_3 is a user specified mitosis cost age multiplier constant.

5.2 Our mitosis detection annealing algorithm

In this section, we present the pseudocode for the annealing algorithm. We first define a function $\textsc{Random}()$, which returns a random double on $[0, 1)$. We also define our temperature function, $\textsc{Temperature}() = \phi_{\max} \cdot (1 - (k/k_{\max}))$ Finally, we define a function $\textsc{ComputeTotalCost}(\text{A})$ which takes a state variable A representing a collection of associations and sums the cost of all associations in the state:

$\textsc{ComputeTotalCost}(\text{A}) =$

$$\sum_{\text{frames } t} \left(\left(\sum_{\text{a}\in \text{ frame t, b}\in t+1,\ a \rightsquigarrow b} \phi(a,b,d_{\max,0}) \right) + \left(\sum_{\text{a}\in \text{ frame t, b,c} \in t+1,\ a \rightsquigarrow b,\ c} \phi_m(a,b,c) \right) \right) \quad (5)$$

5.3 Annealing Results

In this section we present the results of the simulated annealing tracking algorithm for three data sets. As in the previous section, we attempted to link the particles identified by the nucleus identification algorithm across frames by eye, and compared our own links with those links chosen by the algorithm. In addition, we identified likely mitosis events, and compare our mitosis findings with those identified by the algorithm. Table 2 lists the success and failure rates for the annealing algorithm with respect to the decisions made by a human tracker.

Our results suggest that our annealing implementation is slightly less effective than the Sbalzarini tracking implementation in terms of tracking nuclei due to the fact that it does not presently account for disappearing particles[4]. However, the annealing algorithm successfully identifies many mitosis events whereas the Sbalzarini algorithm identifies none. The annealing approach we implemented can easily be expanded to incorporate more elaborate scenarios and even more effective cost functions. In addition, improvements in nucleus identification techniques would help mitigate errors in annealing results; often times, nuclei divide, but the divided particles are identified as a single particle for several frames. Section 6 outlines considerations for future work in these and other areas.

[4]We plan to improve our annealing implementation to account for disappearing particles in future versions

Algorithm 5 Mitosis Detection via Simulated Annealing

Let $A =$ **call** TRACKING()
Set $A =$ **call** LINKING(A)
Let $\phi =$ COMPUTETOTALCOST(A)
Let $k = 0$
repeat
  for all time frames t in $[0 : T-1]$ taken in random order **do**
    // Consider introducing new mitosis events
    for all nuclei a in frame t taken in random order, and nuclei b, c in frame $t+1$ where $a \rightsquigarrow b$ and $\delta \rightsquigarrow c$ **do**
      Let $A' = A$ with $a \rightsquigarrow b, c$.
      Let $\Delta\phi =$ COMPUTETOTALCOST(A') $-\phi$
      if $\Delta\phi < 0$ or RANDOM() $\leq e^{-\Delta\phi/\text{TEMP}(k)}$ **then**
        $A = A'$
        Set $\phi = \phi + \Delta\phi$
      end if
    end for
    // Consider swapping mitosis particles
    for all nuclei a, d in frame t and nuclei b, c, e in frame $t+1$ where $a \rightsquigarrow b, c$ and $d \rightsquigarrow e$ **do**
      Let $A' = A$ when we swap $a \rightsquigarrow b, c$, $d \rightsquigarrow e$ for $a \rightsquigarrow b$, $d \rightsquigarrow e, c$.
      Let $\Delta\phi =$ COMPUTETOTALCOST(A') $-\phi$
      if $\Delta\phi < 0$ or RANDOM() $\leq e^{-\Delta\phi/\text{TEMP}(k)}$ **then**
        Set $A = A'$
        Set $\phi = \phi + \Delta\phi$
      end if
    end for
    // Consider breaking existing mitosis events
    for all nuclei a in frame t and nuclei b, c in frame $t+1$ where $a \rightsquigarrow b, c$ **do**
      Let $A' = A$ when we break association $a \rightsquigarrow b, c$ and introduce $a \rightsquigarrow b$ and $\delta \rightsquigarrow c$
      Let $\Delta\phi =$ COMPUTETOTALCOST(A') $-\phi$
      if $\Delta\phi < 0$ or RANDOM() $\leq e^{-\Delta\phi/\text{TEMP}(k)}$ **then**
        Set $A = A'$
        Set $\phi = \phi + \Delta\phi$
      end if
    end for
  end for
  $k = k + 1$
until $k \geq k_{max}$

Table 2: Annealing Results

Data set	# Frames	Correct links (% Links)	Incorrect links (% Links)	Correct Mitosis Events (% Mitosis Events)	Missed Mitosis Events (% Mitosis Events)	Mitosis False Positives (% Links)
Sample 1	7	112 (98.25%)	1 (0.88%)	3 (75.00%)	1 (25.00%)	1 (0.88%)
Sample 2	12	128 (98.46%)	2 (1.54%)	2 (100.00%)	0 (0.00%)	0 (0.00%)
Sample 3	21	189 (98.95%)	2 (1.05%)	2 (66.67%)	1 (33.33%)	0 (0.00%)

6 Conclusion and Future Work

Most of the errors in the results obtained by both the Sbalzarini and annealing algorithms arise out of ambiguity due to problems in nucleus identification. We speculate that additional work in three-dimensional parameter refinement (see section 2.3) could help identify nuclei which overlap in two dimensions. A simple "trial-by-error" heuristic could suffice to identify collided nuclei; the software need only postulate that multiple particles exist at each detected particle location, insert a particle to test, and then perform parameter refinement to determine whether or not inserted particles better explain the model.

Both our Sbalzarini and mitosis cost functions do not take into account all of the parameters that we solve for in particle refinement. We could trivially expand our cost functions to incorporate z (depth) coordinate information. In addition, we could incorporate differences in size and intensity of nuclei into the cost functions.

In spite of these limitations, our results demonstrate that annealing is an effective approach for identifying mitosis events in multi-nucleated cell data. Thus, our software will be a valuable asset to biologists hoping to nucleus lineage data in order to better understand the mechanisms regulating cell division.

References

[1] A. S. Gladfelter. Nuclear anarchy: asynchronous mitosis in multinucleated fungal hyphae. Curr. Opin. Microbiol., 9:547–552, 2006.

[2] I. F. Sbalzarini and P. Koumoutsakos. Feature point tracking and trajectory analysis for video imaging in cell biology. J. Struct. Biol., 151:182–195, 2005.

[3] Alexander Barnett. Matlab nucleus parameter fitting implementation (unpublished), 2008.

[4] Michael Thomas Flanagan. Simplex minimization: Michael thomas flanagan's java scientific library. Downloaded 5/20/2008 `http://www.ee.ucl.ac.uk/~mflanaga/java/Minimisation.html`.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. Science, 220(4598):671–680, 1983.