MARKOV CHAIN MONTE CARLO FOR AUTOMATED TRACKING OF GENEALOGY IN MICROSCOPY VIDEOS

KATHLEEN CHAMPION
ADVISORS: ALEX BARNETT, AMY GLADFELTER
DARTMOUTH COLLEGE

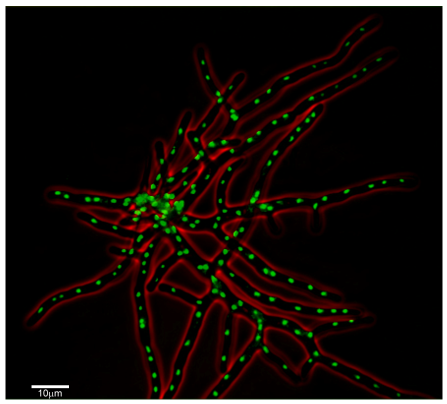
Abstract. Cell biologists use fluorescence time-lapse microscopy to follow the dynamics of proteins in organelles in time and space. Variation in timing during the cell division process can be studied in multinucleate cells by following individual nuclei through time to generate nuclear pedigrees. To undertake a quantitative analysis of mitosis timing, nuclei should be tracked through time over many frames of a time-lapse data set. This is challenging because the images have a low time resolution as well as a low signal to noise ratio, making both tracking and object identification challenging. While methods have been developed for tracking the movement of particles, there are few which have successfully incorporated mitosis to track dividing nuclei. In this paper, we treat the tracking problem as a high-dimensional statistical inference with noisy data and use Markov chain Monte Carlo to sample the posterior distribution. We present cases in 1D and 2D. We also introduce an algorithm for fitting 3D locations given multiple focal planes.

1. Introduction

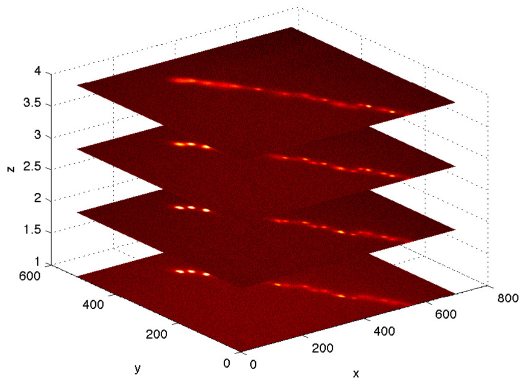
Biologists often wish to study the regulation of mitosis in cells, which has important applications to both infectious diseases and cancer research. The Gladfelter laboratory at Dartmouth College looks at *Ashbya gossypii* cells (figure 1(a)), a multinucleate fungus, in order to learn more about variation in the timing of mitosis. In these cells, division times can vary greatly among nuclei, despite their proximity in the cell and possession of the same genetic code. By studying the division cycles of the *A. gossypii* nuclei on a large scale, the Gladfelter group deduces the factors influencing random variation in cells.[1]

In the Gladfelter lab, hundreds of images of *Ashbya gossypii* cells are captured using fluorescence microscopy. Green fluorescent protein (GFP) can be linked to a protein that localizes in the nuclei, allowing them to be visible under a fluorescence microscope. Currently, these microscopy videos are analyzed by hand, which is a very time-consuming process. The fluorescent light used to image cells will eventually kill them, and so biologists wish to limit exposure by taking images separated by longer time intervals and using fewer slices in the z-direction in order to increase their lifespan. This makes tracking the nuclei more difficult as they will move further between each image, making it less clear what has happened in the time between images. A reliable software tool to automate tracking and determine the most likely explanation of nuclei movement would enable researchers to embark on larger-scale experiments that uncover new correlations.

When biologists wish to track the nuclei, they start with a set of images and generate nuclear pedigrees—information about how the nuclei move and divide between time frames. A useful software package would therefore provide information similar to these hand-generated nuclear pedigrees. Our problem, then, is given $I \in \mathbb{R}^{MP}$, a large vector of all image pixel



(a)



(b)

FIGURE 1. (a) shows an image of the *A. gossypii* cell. (b) shows a stack of 3D microscopy images of the cell at a single instant of time.

intensities at the P pixels, at all of the M time frames, compute $x \in \mathbb{R}^n$ a parameter vector giving the motions of the nuclei in the images and their genealogies.

Evan Tice '09 has already developed some code that aims to solve this problem. His work identified nuclei as peaks in a series of images and expanded on a tracking algorithm by Sbalzarini, et al., in order to identify mitosis events.[4][5] However, the new method for tracking nuclei which we will introduce offers some improvements. In Tice's method, once the peaks are chosen all of the original image information is discarded. This causes problems when peaks are missed in the original pass through, which can happen fairly often. Our new method constantly checks the proposed solutions against the raw image data, and thus is able to better handle more ambiguous information. In addition, Bayesian inference gives us a posterior distribution function over x, not merely a single "best" value of x. This is useful because in some situations multiple parameter sets x can appear equally likely given the data. Humans may also be able to look at multiple values of x given by the posterior distribution function and determine by eye which is actually the most likely.

Our method for tracking nuclei uses the Metropolis-Hastings algorithm, a Markov chain Monte Carlo method, to draw samples from the posterior probability distribution that a particular explanation is correct given the observed data. In Section 2 we explain this method, giving an overview of how the posterior is sampled. Section 3 shows how it works on a toy model in 1-dimension, and Section 4 expands the method to real data and the full 2-dimensional model. Overall this method is computationally intensive (it takes our simple examples about an hour to run for 10^6 iterations of Metropolis-Hastings). However, it is still more useful than having humans track nuclei manually, which is a not only difficult but stressful task.

Section 5 covers a method for fitting nuclei locations in 3-dimensions. We can get 3-dimensional data in the form of a stack of images with different z (see figure 1(b)). Different nuclei are more or less in focus in different z-slices, which allows us to fit for a nucleus's z-location in addition to the x, y-locations. This could be incorporated into the Markov chain Monte Carlo method of tracking in order to clear up difficult problems such as crossing events (crossing in 2D seen in figure 2(b)). Crossing events become easier to explain when 3D locations of overlapping nuclei can be determined. This is because the two nuclei will

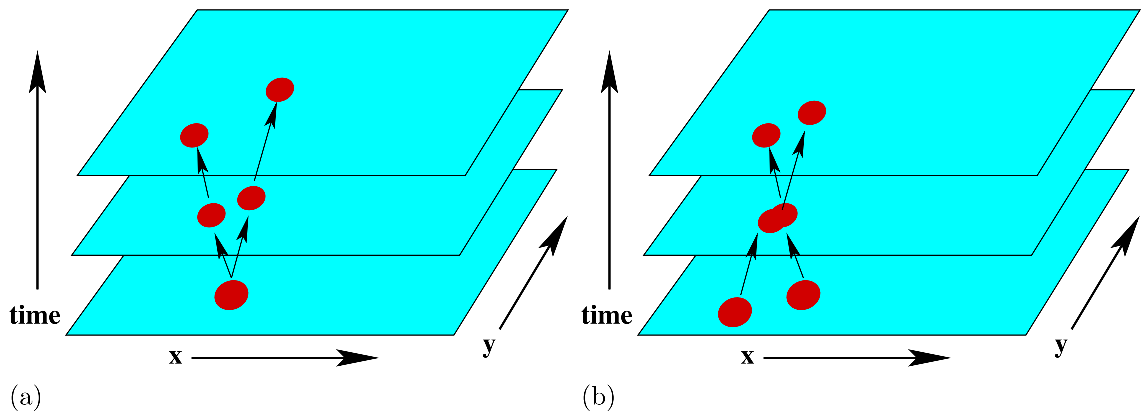


Figure 2. Nuclei in the cell moving in the cell over time. In (a) the nucleus undergoes mitosis between time 1 and time 2. In (b) the nuclei cross between times 1 and 3.

be in different locations in three dimensions (one being above the other), as opposed to having indistinguishable locations, which is what happens when we convert the images to two dimensions using a maximum projection of the three-dimensional image slices. If we can expand our model to determine their 3D positions before and after the overlap as well, we have more information to assist in determining the most likely explanation of the crossing event.

2. Methods for Sampling from the Posterior

2.1. **Bayesian Inference.** Our goal is to draw samples from the distribution known as the posterior distribution. We have a set of parameters $x \in \mathbb{R}^n$ (where n is the number of unknown real parameters) which provide our information about the nuclei's behavior. We treat this set of parameters x as a random variable with a posterior distribution function. The parameters contained in x can be seen in table 1, where n_p is the number of nuclei in our model and t is the number of time frames we are looking at. We can see therefore that n can grow quite high for larger data sets. Even the small examples we work with in Section 4, which for the most part have 4 nuclei and 6 time frames, will have $n = 112$, so $x \in \mathbb{R}^{112}$.

The posterior probability which we wish to draw from is proportional to the product of the prior probability $p(x)$, our prior belief about x, and likelihood $p(I|x)$, the probability of getting the observed data I given x. So given our observations I, the posterior distribution $\pi(x)$ is:

$$\pi(x) := p(x|I) \propto p(x)p(I|x)\ . \tag{1}$$

[2, ch. 3] This gives us not just a single maximum likelihood explanation, but a probability distribution function over all explanations. We therefore would like to find a way to sample from this distribution in order to obtain possible explanations for our data. The prior probability $p(x)$ is equal to a constant, and since we only know $\pi(x)$ up to an unknown normalization factor Z, we can say that the likelihood $p(I|x)$ (which we are able to calculate) is proportional to the prior. In order to calculate likelihood, we start by creating a model image $I(x)$ from x and defining $J(x)$, the sum of squared differences between our original

Parameter	Number of Dimensions	Member of
On/off for each nucleus	n_p	$\{0, 1\}$
Appearance time for each nucleus	n_p	$\{1, ..., M\}$
Disappearance time for each nucleus	n_p	$\{1, ..., M\}$
Parent for each nucleus	n_p	$\{0, ..., n_\mathrm{p}\}$
x,y locations for each nucleus and time frame	$2n_\mathrm{p}M$	$\mathbb{R}$
Widths for each nucleus and time frame	$n_\mathrm{p}M$	$\mathbb{R}$
Peak intensities for each nucleus and time frame	$n_\mathrm{p}M$	$\mathbb{R}$
Total	$4n_\mathrm{p}(1 + M)$	

Table 1. This table shows the different parameters that make up x. As we can see, some of the parameters are discrete and thus x is not actually a member of $\mathbb{R}^n$—however, for the simplicity of our proofs we can treat it as though it is. All the proofs will work in the discrete case as well.

image data I and the model image $I(x)$:

$$J(x) = \sum_{j=1}^{P} \sum_{t=1}^{M} (I_{j,t} - I_{j,t}(x))^2 \tag{2}$$

for pixels j and time frames t. We also define a temperature T, a constant which corresponds to how much noise we expect in the data. A lower value for T will give a higher penalty in $L(x)$ for differences between I and $I(x)$, whereas a higher T means we expect more noise in the data. J is an objective function which we wish to minimize. Now that we have defined these two values, we can define likelihood $L(x)$ as follows:

$$L(x) := p(I|x) = e^{-J(x)/T} \ . \tag{3}$$

So a higher value for J will result in a lower likelihood L. Later, we will also multiply $L(x)$ by penalties for various events happening.

2.2. **Markov Chain Monte Carlo.** Markov chain Monte Carlo methods are a technique which provide a way to sample from a probability distribution by constructing a Markov chain whose equilibrium distribution is the same as the distribution we wish to sample from. In our case, we start with our data I, or our set of images, and we want to draw samples from the posterior probability distribution that a set of parameters x correctly explains this data. We therefore wish to create a Markov chain with equilibrium distribution $\pi(x)$ (as it is defined in equation (1)). This Markov chain will have a transition probability $p(x, x')$, which is subject to a normalization condition

$$\int p(x, x')\,\mathrm{d}x' = 1 \tag{4}$$

for all x.

To obtain the desired Markov chain, we use a Markov chain Monte Carlo method called Metropolis-Hastings. In Metropolis-Hastings, from a set of parameters x, we wish to jump to a new state x' which is not too far from x. Sampling $\pi(x)$ on a Cartesian grid would be difficult because x is in such a high dimension. In fact, sampling from a Cartesian grid would be exponentially hard. For example, if we had only 10 points per dimension the size of the grid would be 10^n. If $n = 100$, the grid is now of size 10^{100}, which is more than the

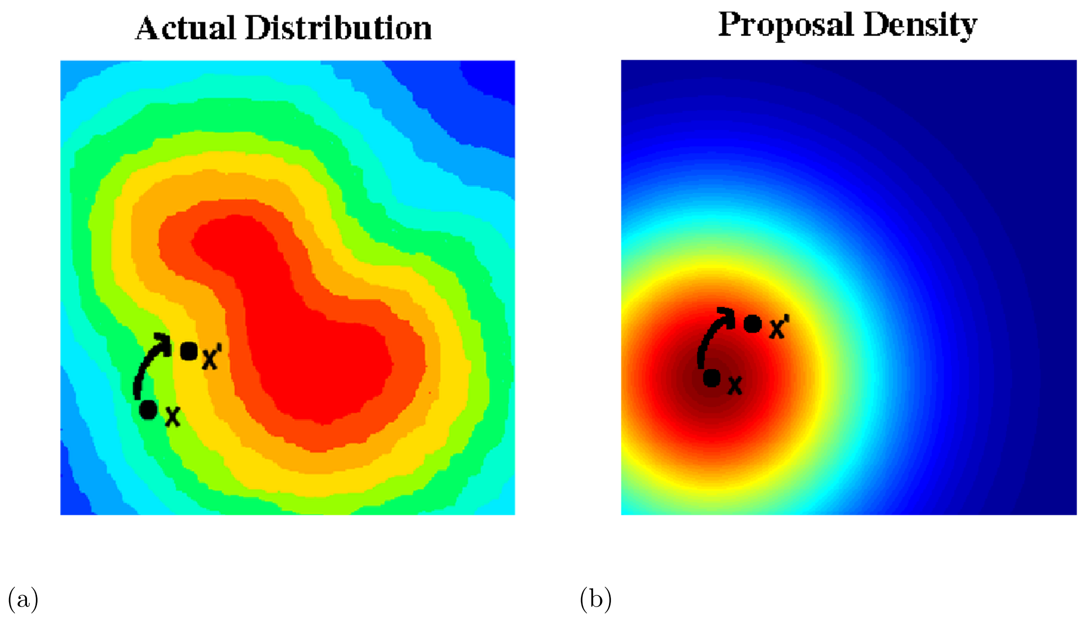


Figure 3. (a) is an example of a distribution $\pi(x)$ we would wish to sample from, with red representing a high probability, blue representing low probability. (b) shows a proposal density $q(x, x')$ for one Markov chain Monte Carlo iteration, which is centered at the current iteration and from which we will draw the next proposed iteration.

number of particles in the universe. Metropolis-Hastings is also preferable in this case to rejection sampling, which generates samples from a proposal density $Q(x)$ and accepts them only if they are also below $\pi(x)$, or importance sampling, which generates samples from a probability density function $Q(x)$ and weighs their importance based on their likelihood in $\pi(x)$ and $Q(x)$. Importance sampling and rejection sampling would have a very small acceptance ratio in a case like this, again because of the high dimension and the fact that $\pi(x)$ is a complicated probability density function.[3]

A Markov chain is called reversible if it satisfies the detailed balance condition

$$\pi(x)p(x,x') = \pi(x')p(x',x) \; . \tag{5}$$

If we want to design a Markov chain with an equilibrium distribution $\pi(x)$, we simply need to set up a chain with a transition probability $p(x, x')$ satisfying equation (5).

Lemma 2.1. *Let $p(x, x')$ and $\pi(x)$ satisfy equation* (5)*. Then π is an invariant distribution, or*

$$\pi(x') = \int \pi(x)p(x,x')\,\mathrm{d}x \; .$$

This is a standard result—we show the proof for convenience.

Proof. We have:

$$\begin{aligned}\int \pi(x)p(x,x')\,\mathrm{d}x &= \int \pi(x)\frac{\pi(x')}{\pi(x)}p(x',x)\,\mathrm{d}x\\ &= \pi(x')\int p(x',x)\,\mathrm{d}x\\ &= \pi(x')\ (\text{by equation (4)})\ .\end{aligned}$$ □

So given a set of parameters x, say we had a Markov chain with transition probability $p(x,x')$ of going from x to x'. We have shown that if $p(x,x')$ satisfies equation (5), then our posterior probability distribution function $\pi(x)$ is an invariant distribution of the chain. As we keep sampling over a large number of iterations, x should diffuse around π. Eventually, after enough iterations, we will be able to draw independent samples from π by spacing them out long enough down the chain.[3]

We create a proposal probability density function $q(x,x')$ in x' localized about x (for example, figure 3(b) shows $q(x,x')$ as a Gaussian centered about x). This is our random state function, which takes a set of parameters x and returns a new set of parameters x' which is only a small change from x. We accept a new state x' (or a new set of parameters) with probability:

$$\alpha(x,x') = \min\left(\frac{\pi(x')}{\pi(x)}\frac{q(x',x)}{q(x,x')},1\right)\ . \tag{6}$$

[3, p. 12] The probability of choosing x' as the new state is $q(x,x')$, and the probability of accepting x' as the new state is $\alpha(x,x')$, so we then get the probability of transitioning from a state x to a state x'

$$p(x,x') = q(x,x')\alpha(x,x')\ . \tag{7}$$

Equations (6) and (7) make the Metropolis-Hastings update rule. If x' is accepted as the new state, x' becomes the new x in the chain. If x' is rejected, x is added to the chain again for another iteration. The states chosen by the Metropolis-Hastings update rule in this manner give us a Metropolis walk along the distribution (see figure 4). These states will give us a Markov chain with transition probability $p(x,x')$.[3] By Lemma 2.1, we know that if this Markov chain satisfies equation (5) then our posterior distribution π will be an invariant distribution of the chain. We wish to show that this is the case.

Proposition. *Metropolis-Hastings (i.e. the update rule equations* (6) *and* (7)*) ensures the detailed balance condition (equation* (5)*).*

This again is a standard result of Markov chain Monte Carlo methods, but we show the proof for convenience.

Proof. There are three possible cases for the relation of $\pi(x)$ and $q(x,x')$. We take a look at each of those cases:

Case 1. If $\pi(x')q(x',x) > \pi(x)q(x,x')$:

$$p(x,x') = q(x,x') \qquad \text{and} \qquad p(x',x) = \frac{\pi(x)q(x,x')}{\pi(x')q(x',x)}q(x',x)$$

So we can show:

$$\begin{aligned}\pi(x)p(x,x') &= \pi(x)q(x,x') \\ &= \pi(x)q(x,x')(\frac{\pi(x')}{\pi(x')}\frac{q(x',x)}{q(x',x)}) \\ &= \pi(x')(\frac{\pi(x)}{\pi(x')}\frac{q(x,x')}{q(x',x)}q(x',x)) \\ &= \pi(x')p(x',x)\end{aligned}$$

Case 2. If $\pi(x')q(x',x) = \pi(x)q(x,x')$:

$$p(x,x') = q(x,x') \qquad \text{and} \qquad p(x',x) = q(x',x)$$

So:

$$\begin{aligned}\pi(x)p(x,x') &= \pi(x)q(x,x') \\ &= \pi(x)q(x',x) \\ &= \pi(x')p(x',x)\end{aligned}$$

Case 3. If $\pi(x')q(x',x) < \pi(x)q(x,x')$:

$$p(x,x') = \frac{\pi(x')}{\pi(x)}\frac{q(x',x)}{q(x,x')}q(x,x') \qquad \text{and} \qquad p(x',x) = q(x',x)$$

So:

$$\begin{aligned}\pi(x)p(x,x') &= \pi(x)\frac{\pi(x')}{\pi(x)}\frac{q(x',x)}{q(x,x')}q(x,x') \\ &= \pi(x')q(x',x) \\ &= \pi(x')p(x',x)\end{aligned}$$

Because these are the only three possible cases and we have shown that equation (5) is satisfied in each case, we have shown that Metropolis-Hastings indeed ensures detailed balance. □

Therefore the Metropolis walk gives us a Markov chain with transition probability $p(x,x')$ and which satisfies equation (5). Convergence is not guaranteed, but our hope is that the Markov chain will converge to the invariant distribution π, which is our posterior probability distribution. If we run Metropolis-Hastings for a sufficient number of iterations, the chain will hopefully converge and we will end up drawing samples from the posterior distribution.

In order to use Metropolis-Hastings, we have defined $\alpha(x,x')$ in terms of $\pi(x)$, $\pi(x')$, $q(x,x')$, and $q(x',x)$ (equation (6)). We can simplify this calculation, however, if we make our proposal density symmetric (i.e. $q(x,x') = q(x',x)$). In other words, the probability of choosing a set of parameters x' starting from x is equal to the probability of choosing x starting from x'. In this case, $q(x,x')$ and $q(x',x)$ cancel to leave us with

$$\alpha(x,x') = \min(\frac{\pi(x')}{\pi(x)}, 1) \ .$$

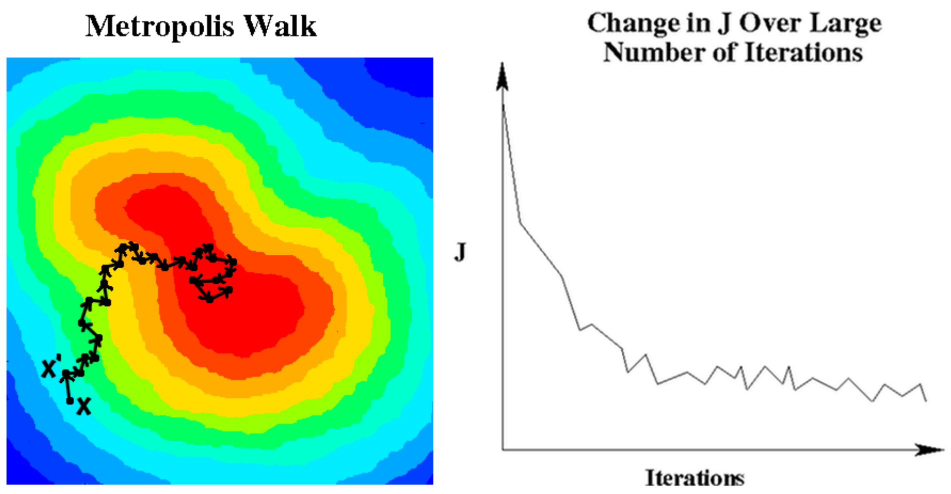


FIGURE 4. The left image is an example of a Metropolis walk along the distribution, starting in the low likelihood area in light blue and moving toward the red area of high likelihood. The right plot is a typical variation in objective function J, the sum of squared differences between the model image and actual image (as defined in equation (2)), over a large number of iterations.

Because π only appears here in a ratio and because we have likelihood L which is proportional to π, we are able to calculate α using L.

2.3. **Autocorrelation.** When sampling from the posterior distribution, we want to make sure that the samples we take are independent from one another. In order to do this, we look at what is called the autocorrelation of the Markov chain. Autocorrelation is a measure of how correlated the chain is with itself at a given time offset τ—thus to ensure that two samples we take are independent, we need to make sure that the autocorrelation $C(\tau)$ over that number of iterations τ is very small (e.g. $|C(\tau)| < 0.2$). So we define $C(\tau)$ as the correlation of a variable at two samples τ iterations apart. If we have a random variable X_i whose mean μ and variance σ^2 are the same over all iterations i, then the autocorrelation $C(\tau)$ is defined as

$$C(\tau) = corr(X_i, X_{i+\tau}) = \frac{E\left[(X_i - \mu)(X_{i+\tau} - \mu)\right]}{\sigma^2} \ . \tag{8}$$

We may estimate the autocorrelation for a single variable X_i in our Markov chain (assuming n iterations), however, with the following equation:

$$C(\tau) = \frac{1}{\sigma^2(n-\tau)} \sum_{i=1}^{n-\tau} (X_i - \mu)(X_{i+\tau} - \mu) \ . \tag{9}$$

We calculate μ by taking the mean of X_i over all n iterations. We do not divide by σ^2 at first, but instead define it as the value which would make $C(0) = 1$. This is because for $\tau = 0$, $C(\tau) = 1$ by definition.

When we plot the autocorrelation for a variable X_i, we look for the τ at which $C(\tau) = 0$ and call this τ_{decay}. τ_{decay} tells us how long between samples we must wait to be sure that we are taking independent samples from the posterior (i.e. the correlation between the two

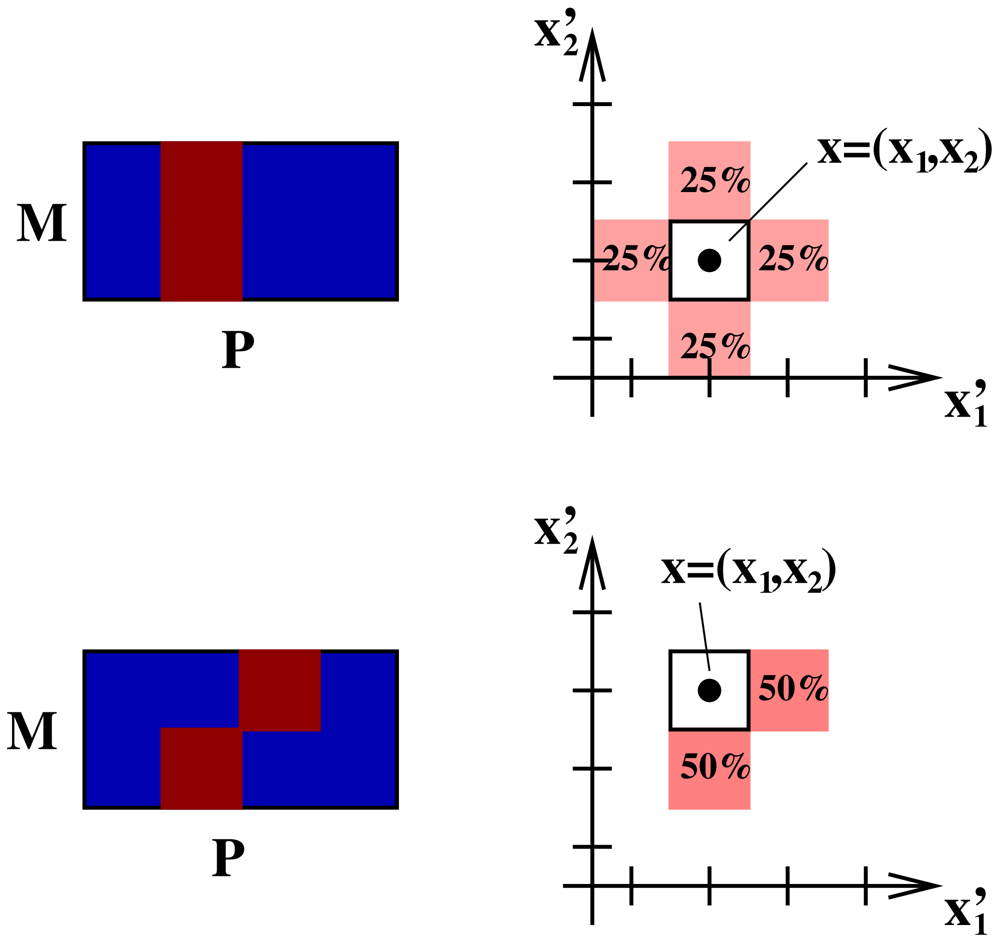


FIGURE 5. Example showing the proposal density $q(x, x')$ for a simple example with $M = 2$ and $P = 4$. In the top case $(x_1, x_2) = (2, 2)$, and in the bottom case $(x_1, x_2) = (2, 3)$. The probability of choosing each pair (x'_1, x'_2) is seen on the right for both cases.

samples is 0). Because we generally only look at one variable to determine τ_{decay}, it is possible that another variable in the chain has a slower decay. However, we usually try to choose a variable X_i which not only is representative of the correctness of our model, but which will have a slower decay than other variables. Thus if we take samples that are separated by τ_{decay} iterations, they will be independent samples from the posterior distribution.

3. 1D Discrete Toy Models

In order to test our methods preliminarily, we start with a toy model of particles moving in 1D. Here the images are an M by P grid with integer values at each site (the number of particles at that location at that time). We let a particle move no more than one position between two time frames. This toy model is much simpler than real image data, as we do not see any variability in the width or intensity of particles.

3.1. **Fixed number of non-splitting particles model.** The simplest example we start with is one particle moving in 1D over time. The number of particles n_p is fixed at 1, and we have positions $x_t \in \{1, 2, ..., P\}$ for $t = 1, ..., M$. This is our parameter vector x.

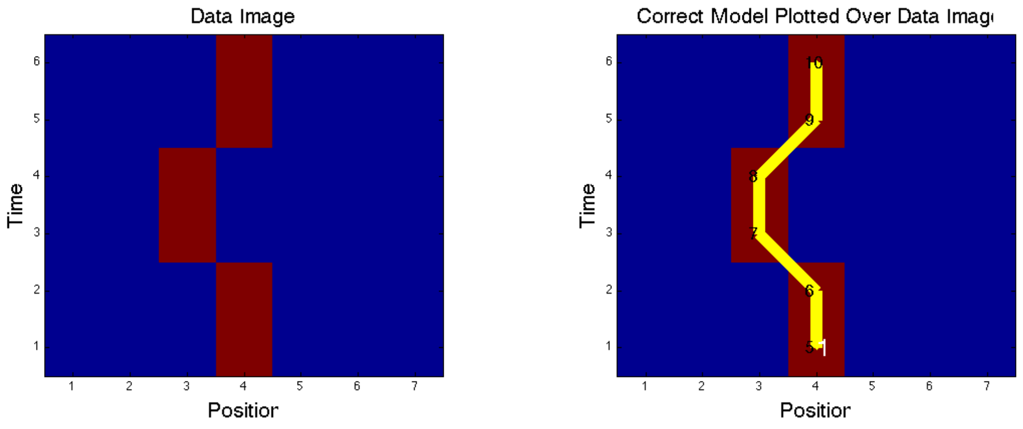
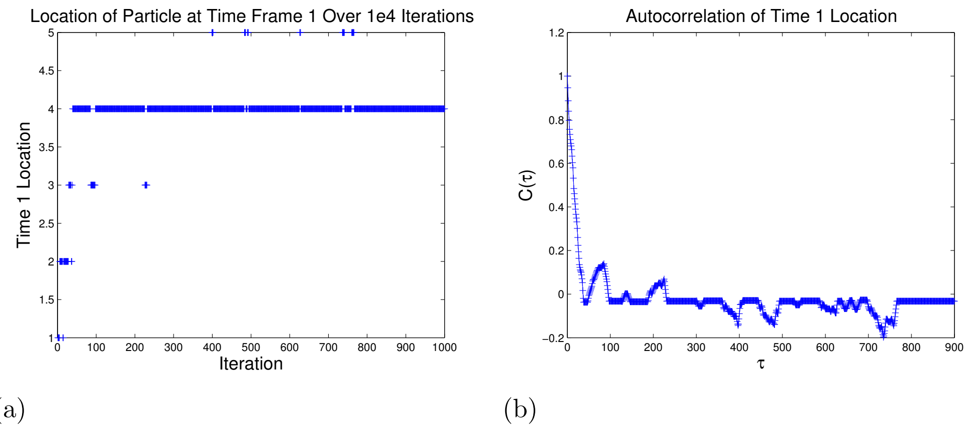
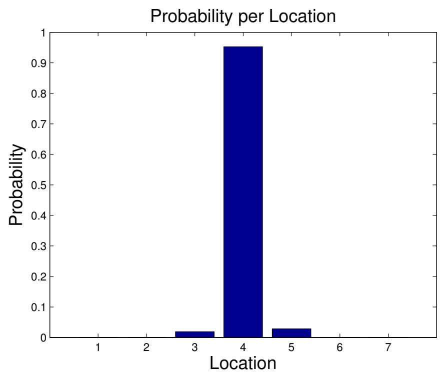


FIGURE 6. On the left is a simple example of a data image, showing a single particle moving in 1D over time. The right shows the plotted model—location of the particle in the model is shown by the line, with the particle's ID number is displayed in white (relevant when there is more than one particle) and the age of the particle at a given time displayed in black (which will be relevant when we add mitosis to the model.

In this example the only Monte Carlo move we use is to modify the particle's location at individual time frames. We forbid moves which create a change in distance of 2 between two time frames (i.e. we must have $|x_{t+1} - x_t| < 2$). In order to carry out a new iteration in the Markov chain, we first choose a time frame t. Then the probability of x_t going to $x_t + 1$ is 1/2 and the probability of x_t going to $x_t - 1$ is 1/2, unless one of these moves is forbidden either by going off the edge of the image or by creating a distance of 2 change between time frames. If one move is forbidden, the other move is always chosen (unless it is also forbidden, in which case we would choose a new t). The proposal density for a full set of moves on an example with $M = 2$ is shown in figure 5. Because there is no mitosis and the particle exists for the entire time, $L(x)$ is defined exclusively by equation (3) (and the rule limiting which moves are legal, which is equivalent to setting $L(x) = 0$ for illegal moves).

In order to evaluate the effectiveness of our method, we must choose some property of the model to evaluate over each iteration. Because this model involves changing only locations, we simply look at the particle's location at a certain time frame (e.g. $t = 1$). Figure 7(a) shows the history of this location over 1,000 iterations, and 7(b) shows the autocorrelation of this value. We look at 7(a) to determine when the chain has "settled" upon the correct solution and is now simply sampling about it—here the settling iteration s appears to be around 50. We use the autocorrelation in order to determine τ_{decay}, or how many iterations we need to wait between samples in order to get independent samples. τ_{decay} is equal to the number of iterations it takes for the autocorrelation to settle to 0, which appears to also be around 50. Figure 7(c) shows the probability of the particle being at each location in P after s. As we can see, after s, x_1 is almost always 4, which is the correct location in the initial data set. If we wanted to draw independent samples from the posterior $\pi(x)$, we would start at the settling time s and take a sample every τ_{decay} iterations. Generally we also need τ_{decay} to be no slower for any other parameters—in this case we can assume this to be true since locations are the only parameters, and there is no reason to expect that this time frame would be different than any other time frames. Therefore the number of independent





(c)

Figure 7. In order to determine when our Markov chain has settled down and how well we have come upon the correct solution, we look at the location of the particle in time frame 1 at each iteration. In (a) we see the history of this location over 1,000 iterations. (b) shows the autocorrelation of this location over these iterations, which needs to settle down to zero so that we can determine τ_{decay}, which appears to be around 50. The settling iteration s, which we can get from (a), also appears to be around 50, and (c) shows the probability of the particle being at each location in time frame 1 from after the settling iteration up until iteration 1,000. This is an estimate of the posterior $\pi(x)$ marginalized onto the single variable x_1 corresponding to the particle's location in time frame 1.

samples s_i we can get with N total iterations is

$$s_i = \frac{N - s}{\tau_{decay}} \tag{10}$$

Thus with $N = 1,000$ iterations, settling time $s = 50$, and $\tau_{decay} = 50$, we can take $(1,000 - 50)/50 = 19$ independent samples.

3.2. **More Complicated Examples and Identity Switching.** Because this simple example has been shown to be successful, we move on to more complicated 1D examples motivated by real-world behaviors that can cause problems in image analysis. Here we allow a number of new Monte Carlo moves because we want to be able to handle more complicated events,

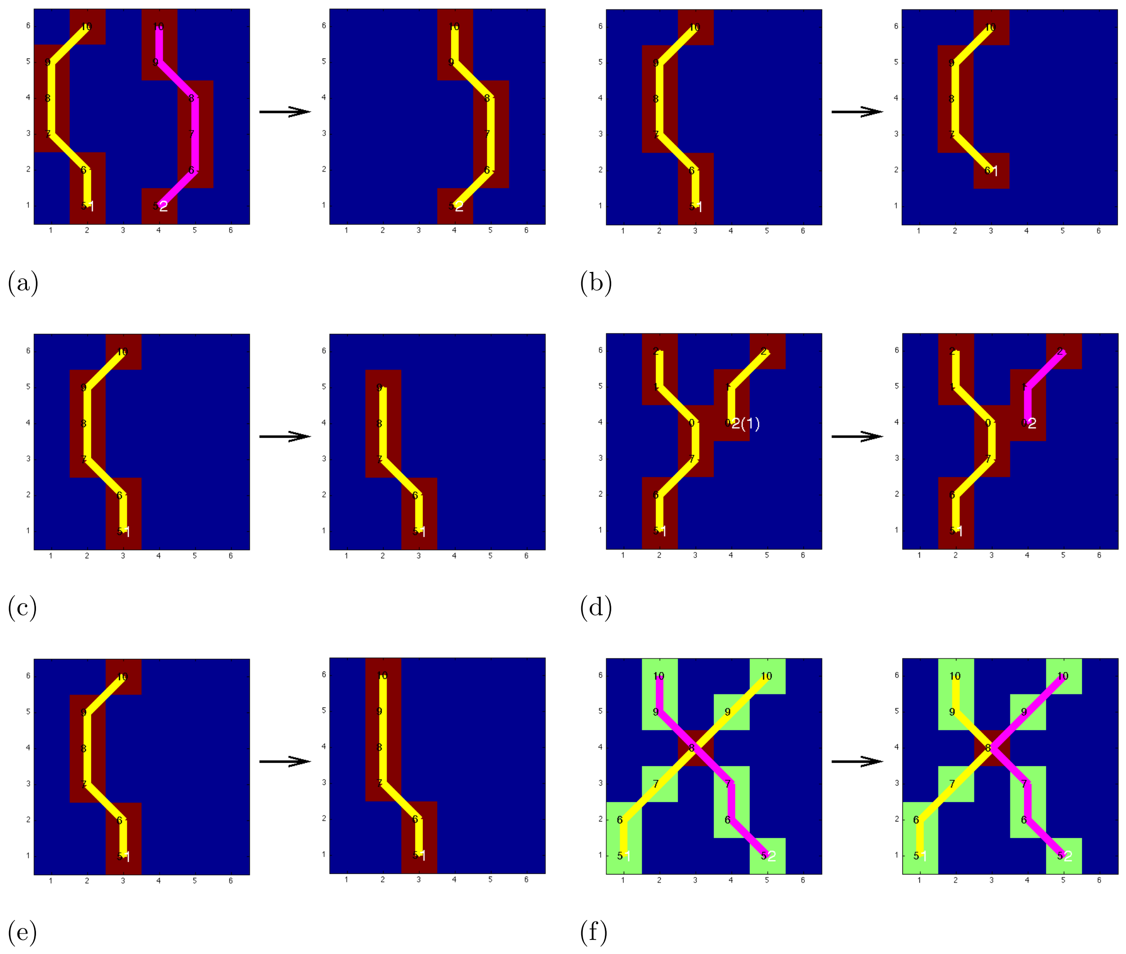


FIGURE 8. Displays the different type of MCMC moves in our random state function. The red, blue, and green (in (f)) boxes are the image data, with the particle tracks implied by parameter x plotted on top (in yellow and magenta). The particles are labeled in white by their number in x (for identification purposes) with their parent if they have one in parentheses. The age of the particle at each time frame is plotted in black (particles which appear in the image not from a mitosis event start their age at T_{inf}, which in this example is 5). (a) shows switching a particle on or off, (b) and (c) show changing the appearance and disappearance times of a particle, respectively, and (d) shows changing a particle's parent. We can also change the location of a particle in any single time frame (seen in (e)). In (f) we see a more complicated move which switches the identities of two particles after a potential crossing event.

such as mitosis and nuclei appearing/disappearing off the edge of the image. Therefore we create the following Monte Carlo moves:

- **On/off** (figure 8(a)): We must allow for the fact that various potential solutions might have a different number of nuclei. Instead of using the more complicated reversible-jump Markov chain Monte Carlo, which allows for a variable number of parameters, we simplify things by simply creating "dummy" nuclei—or having space

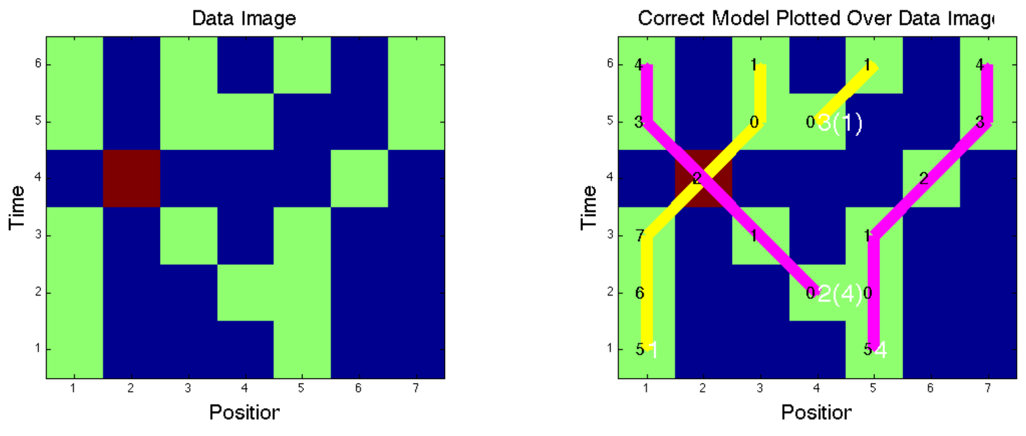
in our model for more nuclei than may actually be in the image—and then letting them be turned "on" or "off". A nucleus which is turned off is not visible in the model image and does not affect the likelihood; however its locations and all other parameters are still there and some of them can be changed in a Monte Carlo move even when the nucleus.
- **Appearance time** (figure 8(b)): Nuclei can appear from the side of the image some time after the start, or they can appear from a mitosis event. Therefore one of the Monte Carlo moves is to change the appearance time of a nucleus by ± 1.
- **Disappearance time** (figure 8(c)): As with appearance, nuclei can disappear off the side of the image before the last time frame. There is a Monte Carlo move to change a nucleus's disappearance time by ± 1.
- **Parent** (figure 8(d)): In order to account for mitosis events, we define each splitting event as having a "parent" and a "child". Biologically, the two nuclei that come from a mitosis event are identical—neither one is the "parent" which gave birth to the other, and instead they are thought of as sister nuclei. However, in order to keep track of nuclei in our model, it is easiest to define a splitting event as a birth event, and so in order to define a mitosis event we can change the parent of a nucleus appearing after time 1 to another nucleus. It is also possible to take a nucleus which has been assigned a parent and assign it to have no parent (which would mean the nucleus did not originate from a mitosis event).
- **Location** (figure 8(e)): The location move is the same as described in section 3.1.
- **Crossing** (figure 8(f)): A complicated situation which we would like to be able to handle easily is when two nuclei cross one another. Switching the identities of two particles which cross at a certain time could take a large number of iterations using the other Monte Carlo moves because it may have to pass through regions of very low probability. Therefore we create a move that identifies a crossing event (a time with two particles in the same location) and changes their identities after the time the crossing event occurred.

The addition of moves such as changing appearance/disappearance times and identifying mitosis events gives us the need to create additional penalties in our calculation of the likelihood, beyond just the difference between the model image and original image. Thus we penalize the following in our likelihood function:

- Nuclei disappearing not near the edge of the image, or appearing not near the edge and not from a mitosis event, $p_{\text{appearance}} = 0$
- A young nucleus (one which has recently undergone mitosis and has age less than T_{inf}, the age we choose at which to stop penalizing splitting) splitting again, $p_{\text{young}} = \sum_i e^{c_{\text{birth}} * \min(0, a_{i,t} - T_{\text{inf}})}$ for i undergoing mitosis ($a_{i,t}$ is the age of particle i at time t;T_{inf} is the age in number of frames at which we stop penalizing a nucleus for splitting, which we choose here to be 5; c_{birth} is another constant which we choose)
- A nucleus must be born near its parent, $p_{\text{pd}} = 0$
- A nucleus must be born before its parent, $p_{\text{pb}} = 0$
- A nucleus's parent cannot be turned off, $p_{\text{po}} = 0$

Setting a penalty to 0 makes an event impossible in the chain. When we introduce these penalties, we get a new formula for likelihood $L(x)$:

$$L(x) = L(x, I) = p_{\text{appearance}} p_{\text{young}} p_{\text{pd}} p_{\text{pb}} p_{\text{po}} e^{-J(x)/T} \ . \tag{11}$$



(a) (b)

Figure 9. On the left is a set of data images for a more complicated 1D example. This example has two mitosis events and a crossing event, which is useful in demonstrating the age feature—young particles should be less likely to undergo mitosis. On the right, (b) shows the correct model plotted over the data image. Lineage information is now included in the plotted model—if a particle split from another, its parent is shown in parenthesis and the two will be plotted in the same color.

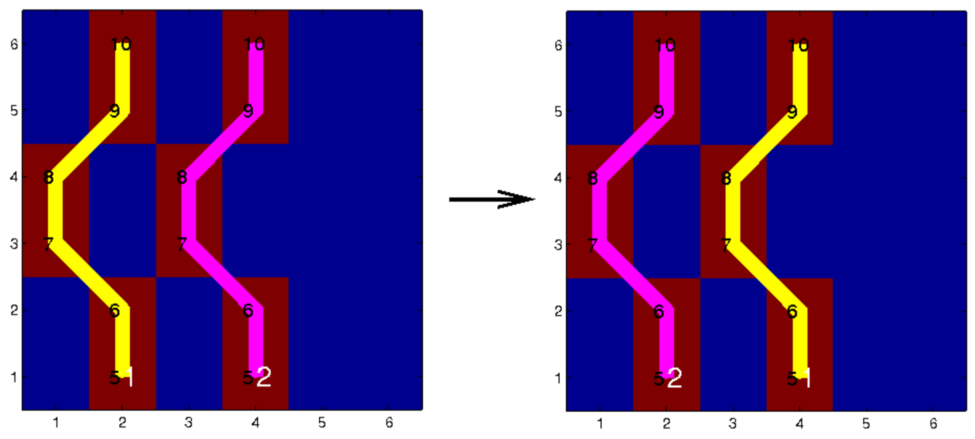
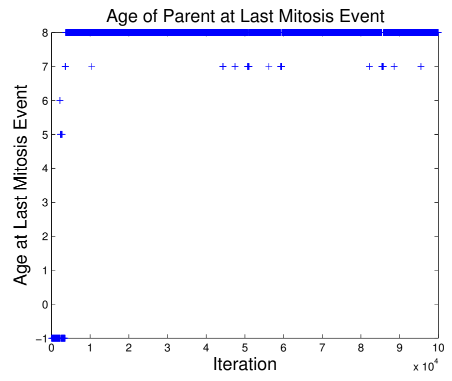


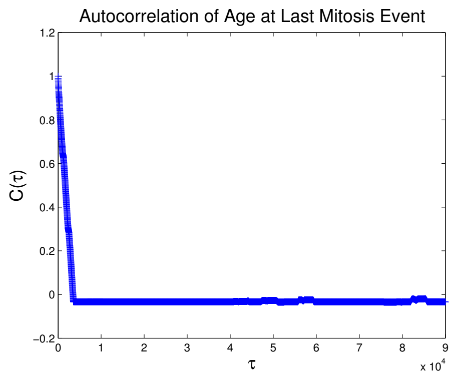
Figure 10. This shows how two particles can switch "identities" in our parameter vector x. Thus when choosing a variable to measure the success, settling iteration, and/or autocorrelation in our model, we must choose a variable invariant to particle identity—e.g. location of leftmost particle at time $t = 1$, instead of location of particle 1 at time 1.

Each of these penalties is dependent on x.

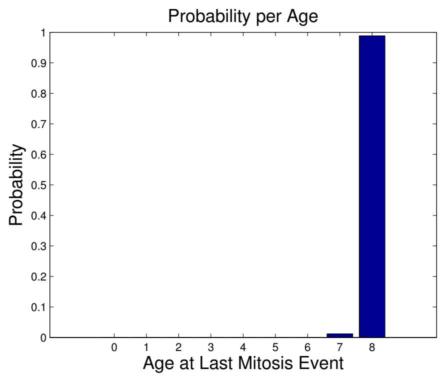
3.3. **Double-Mitosis Example.** We can now introduce a more complicated example in 1D (seen in figure 9) to test the success of our method before moving to real cell images in 2D. We create an example where two nuclei cross, one having split before the crossing and one having split after the crossing. This example will therefore test our penalty for young nuclei



(a)



(b)



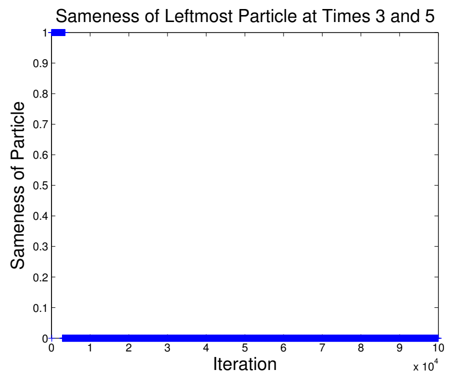
(c)

FIGURE 11. Here we look at the history of the age of the last particle to undergo mitosis in the model from source image seen in figure 9 (with -1 meaning that there was no mitosis event in the model at that iteration). In (a) we see the history of this location over 10^5 iterations. We can see that the chain appears to settle around $s = 5,000$ iterations. (b) shows the autocorrelation of this age over these iterations—here τ_{decay} also appears to be around 5,000. (c) shows the probability of having each age over iterations 5,000 to 10^5—the probability of the age being 8 is almost 1.

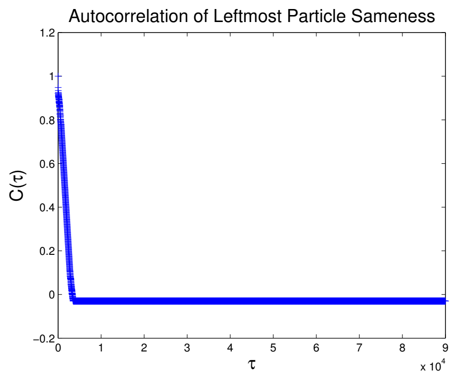
undergoing mitosis. The data image and the correctly plotted model can be seen in figure 9(a) and figure 9(b), respectively.

We again need to choose the variables which we will use to measure the success of our model. Now that we have more than one particle in our model, it is important that these variables be invariant under the particle labeling. This is because particles can switch "identities" in our model. For example in figure 10, particle 1 started on the left but when MCMC is run for 20,000 iterations changing only the locations of the two particles, particle 2 sometimes ends up on the left with particle 1 on the right. Thus it is important that when determining the settling time and autocorrelation of our chain we use variables that are invariant to this, e.g. location of leftmost particle at time 1 instead of location of particle 1 at time 1.

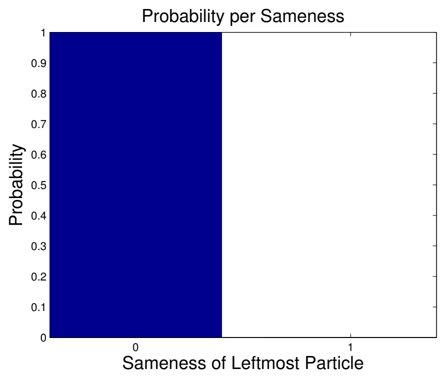
In this example we use a few different factors to measure the success of our model. First we look at the age of the last nucleus to undergo mitosis—this will help us to see if a young



(a)



(b)

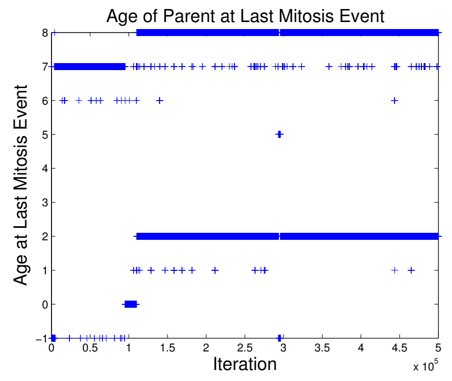


(c)

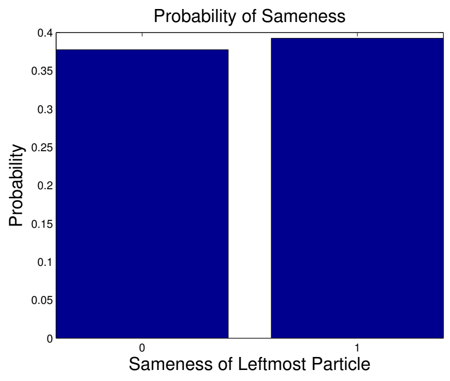
FIGURE 12. Here we look at the history of whether the leftmost particle in the model from the source image seen in figure 9 is the same at time 3 and time 5 (i.e. before and after the crossing event). In (a) we see the history of this location over 10^5 iterations. Here again the chain appears to have settled by $s = 5,000$ iterations—because it is about the same as the settling time seen in figure 11(a), this is a good indication that the whole chain has settled and not solely this individual variable. (b) shows the autocorrelation of this property over these iterations, with τ_{decay} again appearing to be around 5,000. (c) shows the probability of the particle being the same over iterations 5,000 to 10^5.

nucleus is splitting. As we can see, by around 5,000 iterations the Markov chain has settled so that the nucleus undergoing mitosis is almost always age 8. That is also the age of the nucleus splitting in our correct model.

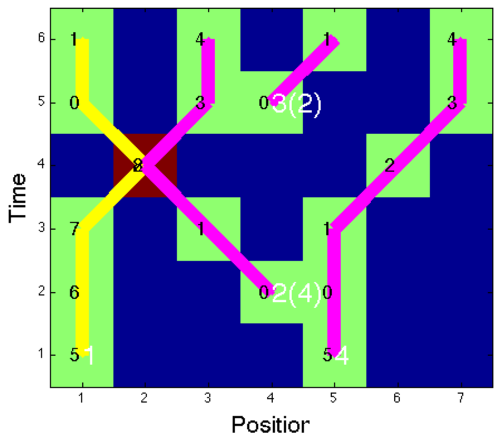
Another factor we can look at is whether or not the nucleus on the left is the same at time 3 and time 5—this will tell us how the crossing event was interpreted. Here the two nuclei at times 3 and 5 are never the same after the chain has settled. This demonstrates that the age penalty is working, because without this penalty it is equally likely that the nuclei did not cross (model shown in figure 13(c). Figure 13 shows what happens when the age penalty is not included—we can see from 13(b) that the probability is about 0.5 that the leftmost particle is the same, and from 13(a) that often a particle undergoes mitosis at age 2. However when we introduce the age penalty the nucleus which split at time 2 is unlikely



(a)



(b)



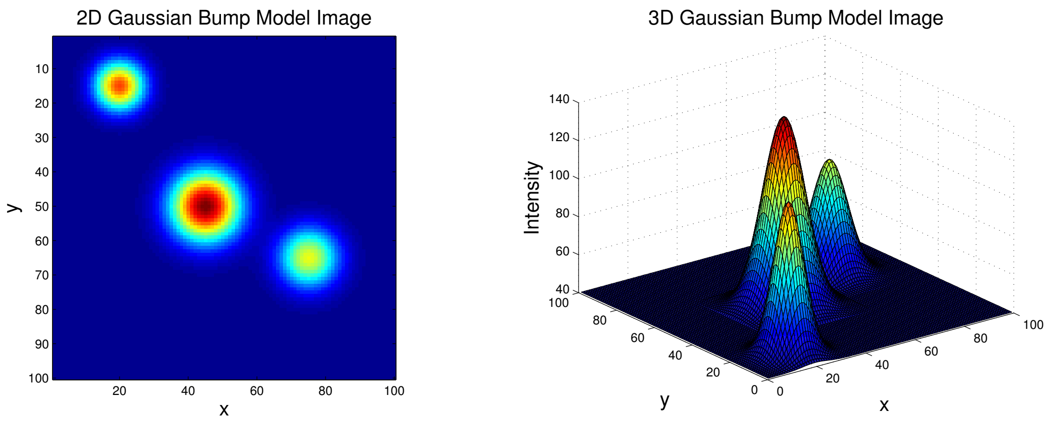
(c)

Figure 13. Here we again look at whether or not the leftmost particle in the model is the same at time 3 and time 5, but in this example the age penalty for mitosis p_{young} is turned off. Looking at the history of the age of the parent at the last mitosis event in (a), the chain appears to settle after around $s = 115,000$ iterations (although it still oscillates between a few values). (b) shows the probability that the leftmost particle in the model is the same at time 3 and time 5 over iterations 115,000 to 500,000. As we can see there is now about a 1/2 probability that the particle will be the same, whereas previously, using p_{young}, there was no chance of this. (c) shows the model of a parameter set x with the crossing event that was previously much more unlikely.

to undergo mitosis again at time 5. This demonstrates the success of our model on this more complicated 1D example.

4. 2D Case

When we expand to 2D in order to incorporate real image data, our model becomes slightly more complex. Instead of using discrete positions in the image, with a 1 representing the presence of a nucleus and a 0 representing no nucleus, a nucleus is now modeled by a Gaussian



(a) (b)

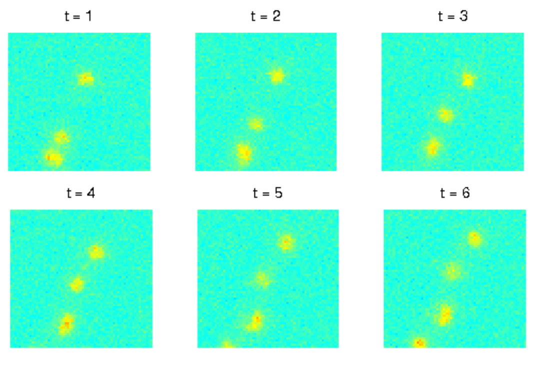
FIGURE 14. (a) is an example of a model image with 3 nuclei represented as Gaussian bumps, shown in 2D. (b) is this same model image plotted in 3D.

bump. Of course nuclei are not Gaussians, but this is a good approximation of how they are seen by the microscope. Figure 14 shows an example of a Gaussian model with 3 nuclei, plotted in both 2D (14(a)) and 3D (14(b)). Each nucleus now has its own width (measured in pixels) and peak intensity in each time frame, which can each be changed as Monte Carlo moves. We also must introduce some new penalties in the likelihood in order to deal with these new parameters:

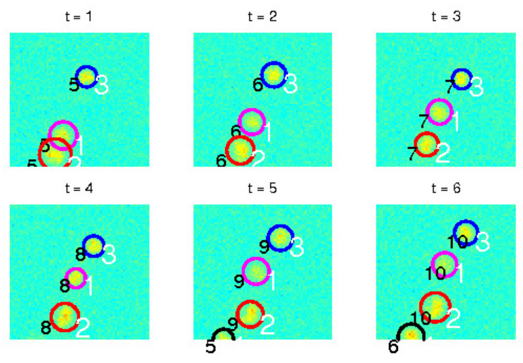
- **How far a nucleus moves between time frames.** In the 1D case, we wouldn't propose a change causing the nucleus to move more than one position between two time frames. However, now the movement is not as discrete, so we use a continuous penalty on how far a nucleus travels with constant c_{dist} and distance traveled d as follows:

$$p_{\text{dist}} = e^{\frac{-d^2}{2*c_{\text{dist}}^2}}$$

  This is calculated for each nucleus between each pair of time frames, and these are all multiplied together in the likelihood. This same equation is used to calculate penalties for the distance a nucleus is from its parent at birth time (p_{pd}), as well as for nuclei which appear/disappear from the edge of the image ($p_{\text{appearance}}$). However in that case we take the distance from the nucleus to the closest edge, with a box of c_{dist} additional pixels on each side, and use the same penalty. We use the box around the image because a nucleus will appear in the image before its center is actually within the edges of the image.
- **How much a nucleus's peak intensity changes between time frames.** It is not likely for a nucleus's peak intensity to change drastically between time frames, unless it undergoes mitosis. When biologists track nuclei by eye, similarity of peak intensities between time frames is one of the factors used to make decisions on how to match up nuclei. Therefore we introduce a constant c_{v} and multiply the likelihood by the following penalty between for each nucleus between each two time frames (except when a nucleus undergoes mitosis), with $v_{i,t}$ as the peak intensity of nucleus i at time



(a)



(b)

FIGURE 15. (a) shows our drifting example raw data images. Three nuclei move over time, and at time frame 5 a fourth nucleus appears from the edge of the image. In (b) we see a successfully fitted model from after our Markov chain settled. This model fits what we determined by eye to be the correct interpretation of the data.

t:

$$p_{\mathrm{v}} = e^{\frac{-(v_{i,t}-v_{i,t+1})^2}{2*c_{\mathrm{v}}^2}}$$

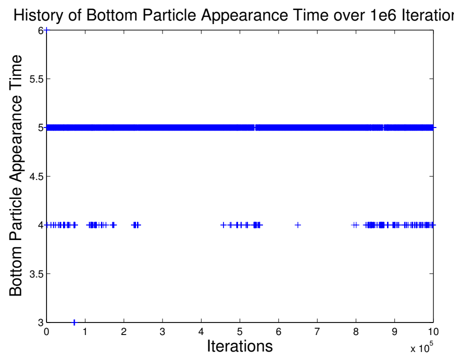
- **Upper and lower bounds on width.** Now that we have a width parameter, we must restrict it to reasonable values only. We therefore make sure a nucleus's Gaussian width is between 1 and 6. We don't penalize changes in width between images, as the width of a nucleus can vary depending on where the nucleus is in the focal plane (i.e. if it moves up and down along the z-axis).

We keep the same penalties p_{young}, p_{pb}, and p_{po} from section 3.2. So our formula for likelihood $L(x)$ is now

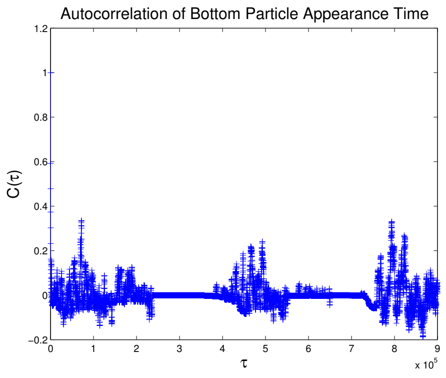
$$L(x) = p_{\mathrm{dist}} p_{\mathrm{appearance}} p_{\mathrm{pd}} p_{\mathrm{v}} p_{\mathrm{young}} p_{\mathrm{pb}} p_{\mathrm{po}} e^{-J(x)/T} \ . \tag{12}$$

In order to test our method on real data, we use relatively simple sets of images with very few nuclei (4 per data set) and time frames (6 per data set). In these image sets we are able to easily identify by eye the correct explanation for the image. Therefore we are able to run our model and evaluate it for correctness. When image sets with more nuclei and time frames are used, it will not be so easy to quickly recognize the correct solution and evaluate the effectiveness of our model.

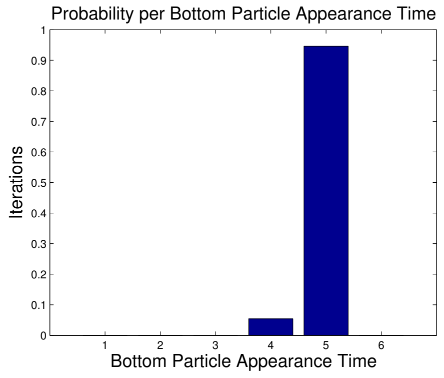
To start, we need to give the data sets an initial guess of parameters x. This should not necessarily fit the correct model, just be somewhat close. In order to get this, we use a simple algorithm which locates peaks (pixels which are local maxima) in all of images and links the closest peaks together between images. In some cases this method finds a set of parameters close to the correct solution, but it will often miss peaks which are too low in intensity and it will not identify mitosis events. Some of the difficulties in this crude algorithm reflect problems with preexisting code such as that by Tice. We give all the nuclei the same default width and no parent, and choose their peak intensity from the original image. From this point on we are ready to carry out the Metropolis-Hastings algorithm, started off from our initial guess.



(a)



(b)



(c)

FIGURE 16. Here we look at the history of the bottom nucleus's appearance time. In (a) we see the history of this location over 10^6 iterations—settling iteration s appears to be around 50,000. (b) shows the autocorrelation of this property over these iterations—we determine τ_{decay} to be around $25,000$. (c) shows the probability of each appearance time for the bottom nucleus over iterations $50,000$ to 10^6.

4.1. **Drifting Example.** The first real data set we look at is a simple example of six time frames with three nuclei drifting over time, and a fourth nucleus appearing in the fifth time frame. The original images are seen in figure 15(a). Here our peak identification method gives us a set of parameters that is very close to the correct solution (the nucleus that appears at time 5 is not identified until time 6). Therefore this example converges very quickly to the correct solution seen in figure 15(b). We use the appearance time of the bottom-most nucleus in time frame 6 as a judge of whether or not we have found the correct solution—this is because this is the most ambiguous piece of information in such a simple example. When we look at the history of this variable over 10^6 iterations, we see that it settles to 5 (which we judge by eye to be the correct appearance time) relatively quickly, but sometimes varying to 4. The actual iteration it has settled by is difficult to determine because it settles so quickly, but we say $s = 50,000$ since it has definitely settled by this iteration. The fact that it is sometimes equal to 4 can be explained by the fact that positioning a nucleus very close to the edge of the image would contribute very little to the difference between the model image

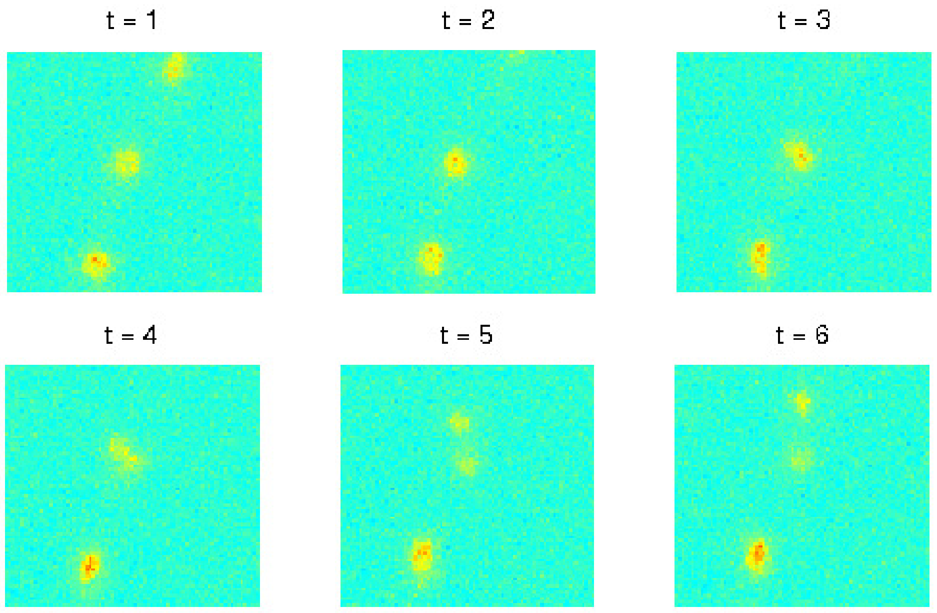


Figure 17. *Mitosis raw data.* Raw data images of nuclei moving over time, with one undergoing mitosis between times 3 and 4.

and the actual image since very little of the nucleus would be seen in the model—thus the likelihood would not be greatly affected.

We see that the autocorrelation for this parameter (figure 16(b)) settles to around zero very quickly as well. We can set $\tau_{decay} = 25,000$ because the autocorrelation appears to have settled to 0 by that number of iterations. Here especially we need to worry about τ_{decay} being no slower for any other parameters in order to ensure that we are actually taking independent samples. We believe this to be true due to the fact that bottom nucleus appearance time is the variable that this model will probably be the most complicated to fit—parameters such as nucleus location, peak intensity, and width will in general have a relatively low τ_{decay}. Thus from equation 10 we calculate that we could obtain $(10^6 - 50,000)/(25,000) = 38$ independent samples. In figure 16(c) we look at the probability of the bottom-most nucleus appearance time being each different value over the iterations after s. We see that it was most often equal to 5—the time which we determine to be correct from looking at the image. If we tightened the extra box width around the image in which we allow a nucleus's center to be contained, we could lower the probability of the nucleus's appearance time being 4. However, in terms of the correctness of our solution, identifying that a nucleus has appeared in the image a few frames early is not of major importance as long as it finds the nucleus at the times when we are able to visually identify the nucleus.

4.2. **Mitosis Example.**

The next example we look at is of more interest—a data set which includes a mitosis event (figure 17). In this example the peak identification does a very poor job of identifying a correct starting point for our Markov chain because many of the peaks are extremely dim (see figure 18). Thus this is an example which benefits from the fact that we continually check against the raw data images—if we discarded our original images after

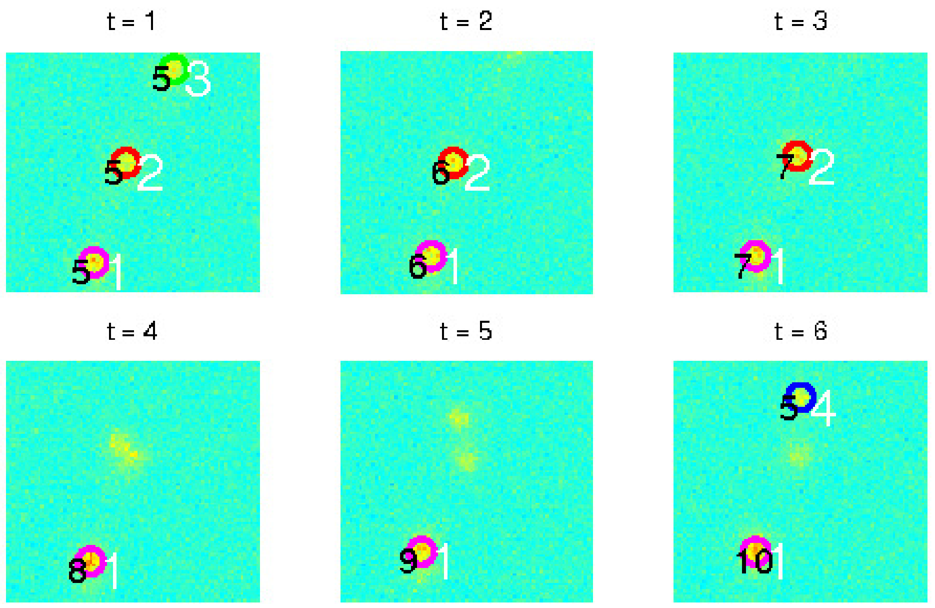


Figure 18. *Initial Guess.* This is the plotted model of our initial guess for x, found from peak identification. As we can see, it fails to identify many of the peaks.

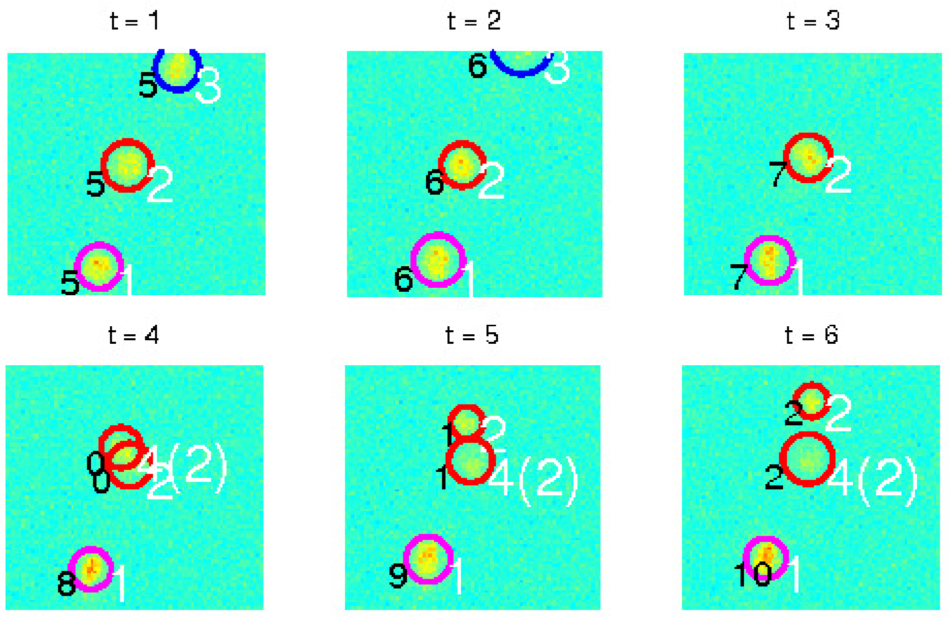
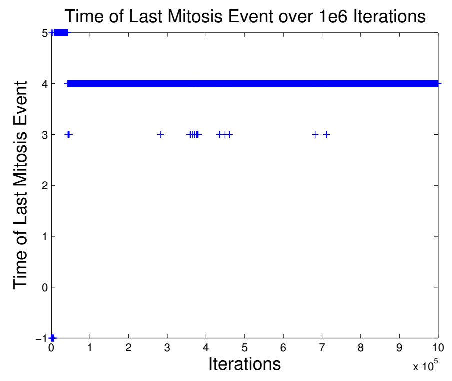
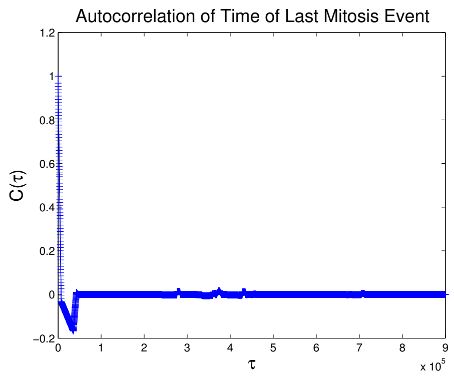


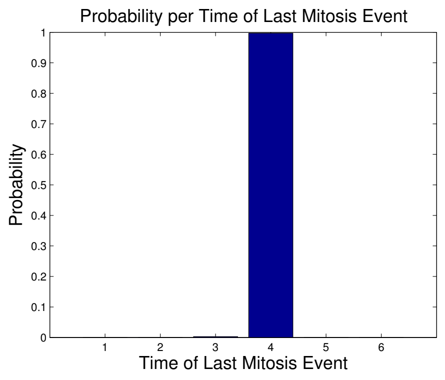
Figure 19. *Successfully fitted model.* This shows a successfully fitted model from after our Markov chain settles. This model is representative of what we previously determined by eye to be the correct interpretation of the data. Two of the nuclei are independent and don't undergo mitosis, with one disappearing from the image at time 3. One nucleus undergoes mitosis between time frames 3 and 4.



(a)



(b)

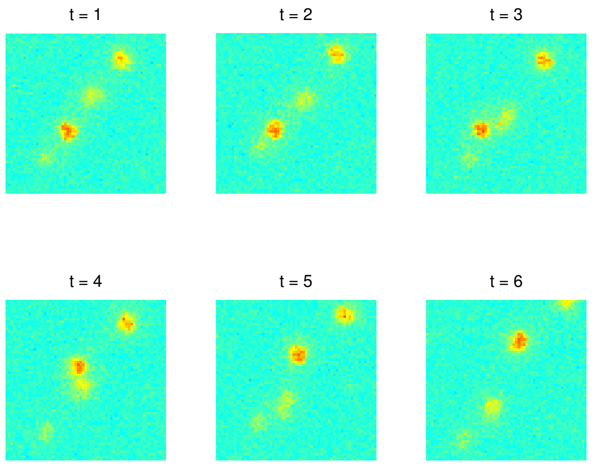


(c)

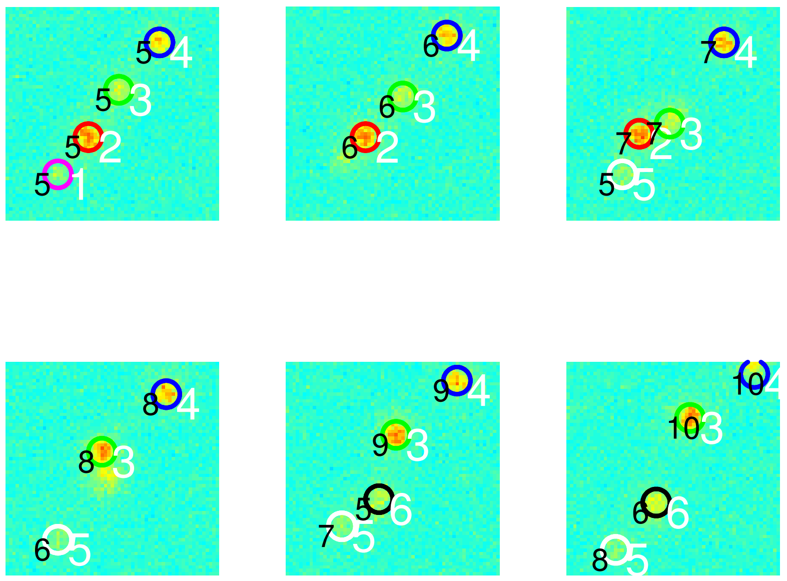
FIGURE 20. Here we look at the history of the time mitosis occured. In (a) we see the history of this location over 10^6 iterations. It appears to have settled to the correct solution by 50,000 iterations, so we say $s = 50,000$. (b) shows the autocorrelation of this variable—τ_{decay} also appears to be around 50,000. (c) shows the probability of the mitosis event occurring at each time over iterations 50,000 to 10^6.

locating the peaks, much of our information would be lost. The correct solution which we have determined by eye is seen in figure 19.

Despite the vast difference between the original model found by peak picking and the correct explanation for the data, our Markov chain converges to the correct solution relatively quickly. The factor we use to measure the correctness of our model is the time at which the last mitosis event occurs (should be the only mitosis event). If no mitosis event occurred, this is represented on our plot as -1. Figure 20(a) shows that by iteration 50,000 the mitosis event is consistently identified as occurring at time 4—thus we set $s = 50,000$. There are some iterations which have mitosis occurring by time 3, but from figure 20(c) we see that the probability of it occurring at time 4 is extremely close to 1. Figure 20(b) shows the autocorrelation of the time of the last mitosis event, with $\tau_{decay} = 50,000$. Again we can use equation 10 to calculate that we are able to obtain $(10^6 - 50,000)/50,000 = 19$ independent samples from this example.

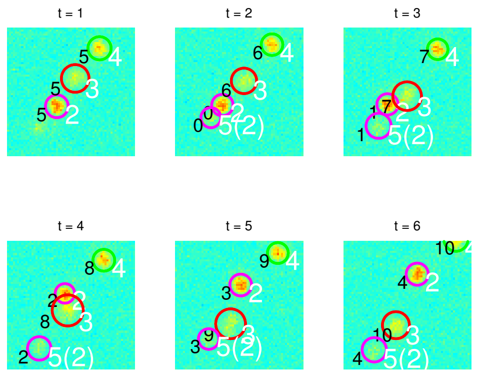


(a)

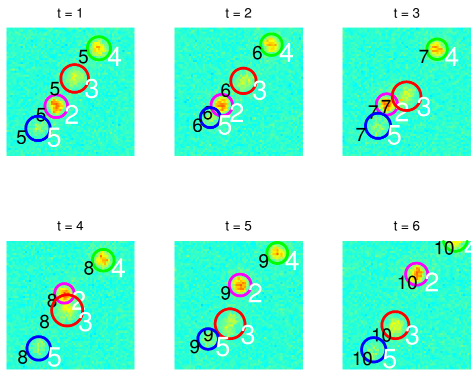


(b)

Figure 21. (a) shows the raw data images for our crossing example. Four nuclei move over time, with two nuclei coming near and passing each other between time frames 3 and 4. In (b) we see the initial guess for our parameters x made by peak identification. As in the mitosis example, this also fails to identify some peaks in the images, causing our initial guess to be very far from the correct explanation of the image data.



(a)

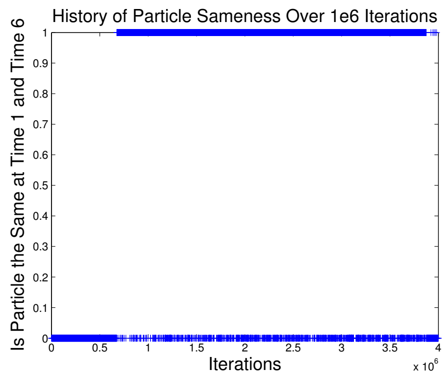


(b)

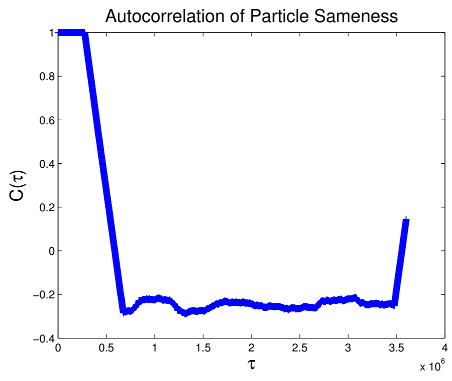
FIGURE 22. (a) is a model that is sampled from Metropolis Hastings. We can see that a peak at time frame 1 is not identified and should actually be matched up with nucleus 5, so this model is not correct. (b) shows a correct interpretation of the data (which we have identified by eye)—our method fails to find this as a solution. However, our method did interpret the crossing event successfully—the high-intensity nucleus which we identified as being the same throughout the time frames is nucleus 2 in this model.

4.3. **Crossing Example.** As stated previously, crossing is another complicated event that is frequently encountered when tracking nuclei. When two nuclei come together and overlap in a certain time frame, humans interpreting the data by eye must use factors such as peak intensity to decide what happened to the nuclei after the crossing event. We look at another set of images which has exactly this situation—two nuclei in the images, one of which is much brighter than the other, come close together and then separate. We use the intensity here to guess the correct interpretation of this data, seen in figure 22(b).

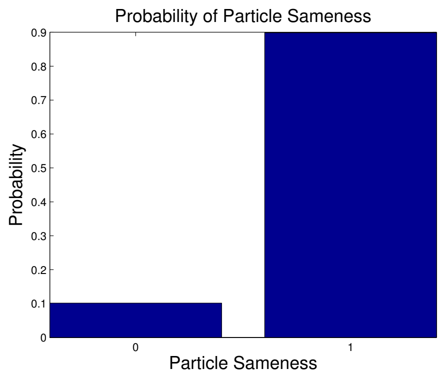
Unfortunately, when we run our Metropolis-Hastings algorithm on this example, we see that the Markov chain fails to settle on the correct interpretation. Instead it settles on the interpretation shown in figure 22(a). This model is mostly correct, but for the mitosis event seen between time frames 1 and 2. There is a clearly visible peak in time frame 1 which is not identified by the model—nucleus 5 should actually have an appearance time of 1 (one frame earlier) and be present at this peak, rather than splitting from nucleus 2. There are several reasons why the Markov chain may have had trouble settling on the correct interpretation seen in figure 22(b). The peak in the first image is relatively dim, so it may only make a small contribution to $J(x)$ even though no peak is present the $I(x)$. In addition, going from the model in figure 22(a) to the model in figure 22(b) would require at least two distinct steps—changing nucleus 5's appearance time and changing its parent. Either intermediate would have a much lower likelihood than these two models, as we penalize a particle appearing in the middle of the image without a parent and also penalize a mitosis event occurring in time frame 1 (because we have no information about the nuclei before this time frame and therefore have no reason to suspect that two nuclei in time frame 1 came from the same mitosis event). This is one of the problems that can occur when sampling the posterior using Metropolis-Hastings—a region with high probability may be surrounded by regions



(a)



(b)



(c)

FIGURE 23. We evaluate success in some sense by determining whether one of the nuclei involved in the crossing event is identified by the model as being the same at time 1 and time 6. (a) shows this history over 10^6 iterations (with 1 meaning the nucleus was identified as the same at the two time frames, and 0 meaning it was different). By 10^6 iterations it appears to be identifying the nucleus as the same most often than not, so we say the settling iteration $s = 10^6$. (b) shows the autocorrelation of this variable—unfortunately C_τ is not as close to 0 as we would like it (we see it is generally between -0.2 and -0.3). (c) shows the probability of the mitosis event occurring at each time over iterations 10^6 to 4×10^6—the probability of the nucleus being the same is about 0.9.

with very low probability, and thus it is difficult for our Metropolis walk to find the region of higher probability. Running Metropolis-Hastings for a higher number of iterations may help solve this problem; however in this example even after 4×10^6 iterations we had not converged to the correct solution.

Despite the fact that we have not converged to the correct interpretation, it is close enough that we are still able to analyze how successful our model was at interpreting the crossing event. The variable we choose to evaluate is whether or not the bright nucleus (nucleus 2 in figure 22) is identified as being the same nucleus in time frames 1 and 6. The initial guess for x was far from any correct interpretation of the images, and we can see it takes over 5×10^5 iterations for the movement of this nucleus to be correctly identified. However, once

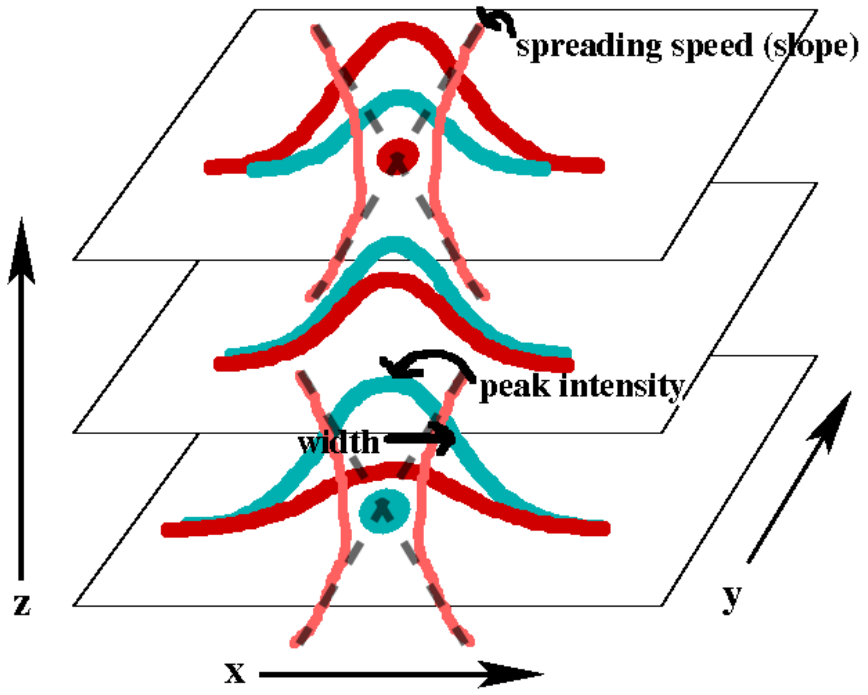


FIGURE 24. This figure shows an example of two nuclei which are overlapping in the x, y-plane but have different resolutions in z. We show an example of how the Gaussian for each nucleus might change for different z-slices, and provide a visual of the width and peak intensity for a nucleus, as well as the spreading speed of light in the example image.

the nuclei at these two time frames are proposed as being the same, the model most often interprets this variable correctly—the probability over the last 3×10^6 iterations of the nuclei being the same was over 0.9.

5. 3D Tracking

One of the major problems that can occur in 2D tracking come about when two nuclei are on top of each other. Not only may this make it difficult to recognize if two nucleui are present rather than just one, but it can also cause problems in the interpretation of crossing events. Fortunately, microscopes enable us to obtain 3D stacks of images. In these image stacks, nuclei appear more or less in focus in different z-slices depending on their location in 3D. Our goal is to use these 3D image stacks to identify nuclei as overlapping and fit them in the 3D plane. The same limitations occur here as in the tracking case—because we are working with live cells, we need to limit the number of z-slices we take (just as we need to take images over longer time intervals when tracking).

Our problem now is, given a stack of images $I \in \mathbb{R}^{XYZ}$, a large vector of Z images each of size XY pixels, compute $p \in \mathbb{R}^n$ a parameter vector containing information about the images and the nuclei in the images.

Parameter	Number of Dimensions
x-location x_i for each nucleus	n_{p}
y-location y_i for each nucleus	n_{p}
z-location z_i for each nucleus	n_{p}
Width w_i for each nucleus	n_{p}
Peak intensity v_i for each nucleus	n_{p}
Background intensity i_b	1
Spreading speed of light s	1
Total	$5n_{\mathrm{p}} + 2$

TABLE 2. This table shows the different parameters that make up p for fitting nuclei in a 3D image frame (single time frame). We see $p \in \mathbb{R}^{5n_{\mathrm{p}}+2}$ where n_{p} is the number of nuclei in the image.

5.1. **Approach.** In order to locate nuclei in three dimensions, we start by creating a model for cell images. We again have a parameter vector $p \in \mathbb{R}^n$ which provides information about the image stack and the nuclei in the image. The parameters contained in p can be seen in table 2, where n_{p} is the number of nuclei in our model. We also define the physical distance between z-slices μ_I, but this is not fit for and is therefore not a part of p.

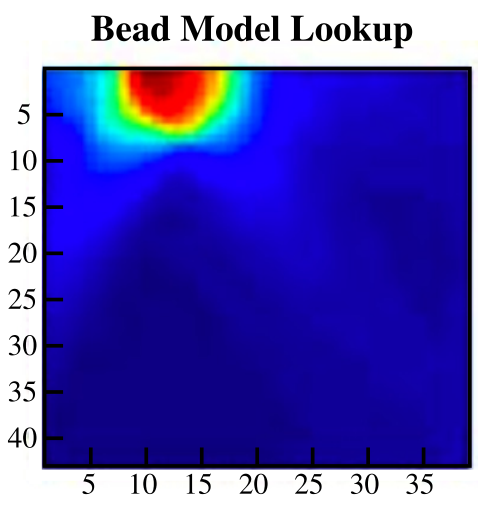
We use p to create a model image $M(p)$ of the same size as I (and with every pixel $M_{x,y,z}(p)$ the same pixel as $I_{x,y,z}$ inside the square). Because our goal is to fit p to the original image data I, we want $M(p)$ to be as close to I as possible. We therefore define an objective function $J(p)$:

$$J(p) = \sum_{(x,y)\in m} \sum_{z} (I - M(p))^2 \quad . \tag{13}$$

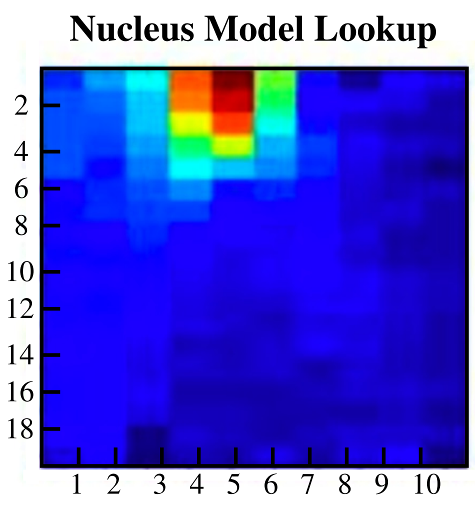
This function does not sum over all (x, y)—we define a mask m, or set of pixels (x, y, z) over which to take this difference. We fix a value w_b, and m will be all the pixels within radius w_b of the center (x_i, y_i) of any nucleus i. The main purpose of the mask is to increase computational speed by limiting the number of pixels we sum over since the optimization must do many evaluations of J. The mask also limits fitting only to the neighborhood of the nuclei—it is not useful to fit the noise elsewhere in the image as it does not give us much new information and may cause us to over fit for the background intensity. Our goal is to minimize $J(p)$, which is a representation of the "error" in our model $M(p)$. In order to do so, we use unconstrained nonlinear optimization (using the function *fminunc* in Matlab).

In order to successfully fit the data in such a way that we can get information about the nuclei in the image, we must find a way to create our model image $M(p)$ from a parameter vector p. We test out a few different methods for creating a model:

- **Gaussian.** In this technique, every nucleus is modeled as a radially-symmetric Gaussian bump in each z-slice with height (peak intensity) and width varying among z-slices according to z and the spreading speed of light s.
- **Bead Lookup.** We are able to take images of beads similar in size to *A. gossypii* nulcei with over 40 z-slices. We use an image of a bead collected in the Gladfelter lab to create a lookup table of intensity at varying radii and z-distance from the center of the bead. We then use this lookup table to determine the intensity of our model at points around each nucleus in the model image.



(a)



(b)

FIGURE 25

- **Nucleus Lookup.** This method for creating a model image is similar to the bead lookup method, except that instead of using a bead image to create the lookup table we use an image of an actual *A. gossypii* nucleus collected in the Gladfelter lab.

5.1.1. *Gaussian Model.* To make a Gaussian model image, we start by setting every pixel in $M(p)$ to our background intensity i_b. We model each nucleus as a Gaussian bump in each z-slice and add these values to $M(p)$. Each nucleus i has an overall peak intensity v_i and width w_i in pixels, but as stated previously, the peak intensity and width for the nuclei's Gaussian representation at each z-slice will vary according to the spreading speed of light and the z-location of the nucleus. We calculate width $w_{i,z}$ and peak intensity $v_{i,z}$ of nucleus i at z-slice z as follows:

$$w_{i,z} = \sqrt{w_i + s(z - z_i)^2} \tag{14}$$

$$v_{i,z} = v_i \left(\frac{w_i}{w_{i,z}}\right)^2 . \tag{15}$$

The value w_b which was used in the mask m in equation (13) is the half-width in pixels of the square around a nucleus's x, y-locations for which we will add the nucleus's Gaussian to the model image. In other words for each nucleus i, for each $(x, y) \in \{x_i - w_b, ..., x_i + w_b\} \times \{y_i - w_b, ..., y_i + w_b\}$ for each z, we perform the following:

$$M_{x,y,z}(p) = M_{x,y,z}(p) + v_{i,z} e^{-\left(\frac{(x-x_i)^2+(y-y_i)^2}{2w_{i,z}^2}\right)} . \tag{16}$$

After performing this iteratively for every nucleus i in p, we get our model image $M(p)$.

5.1.2. *Bead and Nucleus Lookup.* For the bead lookup model, we started with an image of a bead taken over 40 z-slices. We used the Gaussian model to fit for the center of the bead (x, y) as well as the z-location z, width w and peak intensity i_p. Background intensity i_b of the bead image is also fit for, so we subtract i_b from the bead image in order to normalize to a background intensity of 0. In addition, we know the actual distance μ between z-slices in the bead image stack. Once we know the center of the bead, we create a table of the average intensity at each z-slice of all pixels with x, y-Euclidean distance from the center in $\{[0, 1), [1, 2), ..., [42, 43)\}$. Thus for any $z \in [1, 40]$ and any radial distance $r \in [0, 43)$, we define $L(r, z)$ as the intensity in the lookup table in row r (radii in $[r - 1, r)$) and column z (intensities in z-slice z).

To create a model image $M(p)$, we again start by setting every pixel in $M(p)$ to the background intensity i_{b_I} defined in p. We again have w_b, which will work in the same way it did for the Gaussian model. So for each nucleus i, for each $(x, y) \in \{x_i - w_b, ..., x_i + w_b\} \times \{y_i - w_b, ..., y_i + w_b\}$ for each z, we calculate the Euclidean distance r of (x, y) from (x_i, y_i) and add to the model image

$$M_{x,y,z}(p) = M_{x,y,z}(p) + L\left(r\frac{w_i}{w}, z_i\frac{\mu_I}{\mu}\right)\frac{v_i}{i_p}\ . \tag{17}$$

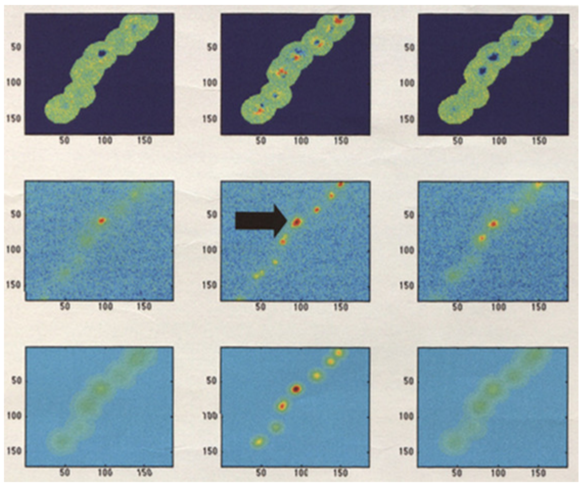
Iterating over every nucleus i in p will give us our model image $M(p)$.

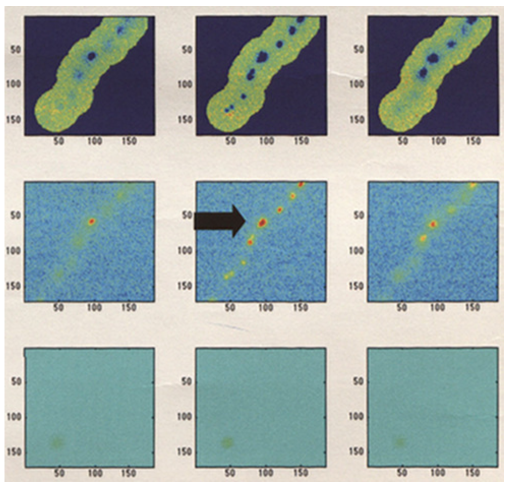
The nucleus lookup model works in the same way as the bead model, except that we use an image of a nucleus instead of a bead. We again use the Gaussian model to fit for information about the nucleus image. The nucleus lookup table is calculated for 10 z-slices and pixels within a radius of 18 from the center of the nucleus.

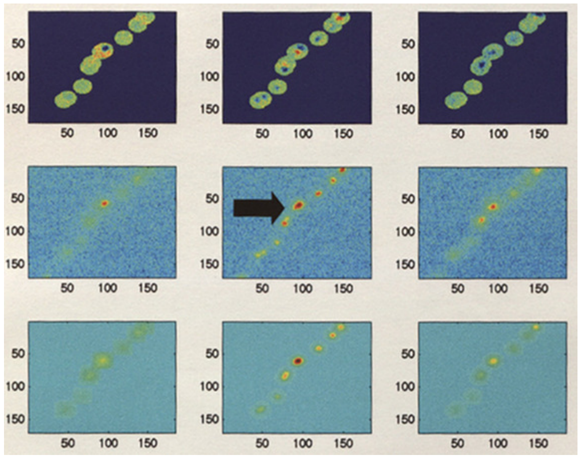
5.2. **Results.** All three of our models were tested on a set of images with 10 nuclei and 3 z-slices. In these images, we can see by eye that one pair of nuclei overlapped but were in focus in different z-slices. Thus we are able judge the correctness of a model not only by the value of the objective function $J(p)$, but also by whether or not these two nuclei were resolved at different z-locations. When fitting with all three models, we started p off by picking (x, y) locations by eye and starting all z-locations in the center of the image stack. Peak intensities were started at the image intensity at (x, y) and widths were all given the same value. Background intensity was set to the average over all pixels in the data images, and we chose an arbitrary starting value for spreading speed s.

When we first fitted using the Gaussian model, it failed to create an accurate model of the image data. The fitting would determine the two overlapping nuclei as being located at about the same z-location, thus failing our test to determine success of the model. This failure provided motivation for creating the bead lookup model, which we hoped would more accurately represent a nucleus in an image than a Gaussian bump. The bead model provided a very close fit (low objective value $J(p)$) for other images of beads, showing that there were no inherent problems with the model. However, when fitting the same nuclei this model performed very poorly. It resulted in a large value for $J(p)$ and failed not only to resolve z-locations but even the peak intensities of nuclei in the image. This model may have yielded such poor results due to the difference in shape between beads and nuclei, as well as size differences. Again with the nucleus model, the model was able to fit for itself (showing there were no inherent problems) but failed to successfully resolve the z-locations of nuclei in our test images. Nuclei have widely varying widths, which might be a reason why this model failed.

Upon revisiting the Gaussian model, we adjusted the value w_b, the half-width of the box surrounding each nuclei for which we calculate the model, and the radius of the mask which we use in equation (13) to calculate $J(p)$. This time the model appears to succeed—fitting resolves the two overlapping nuclei at different z-locations despite having both started off at the same location. The model image appears much closer to the original image data for


(a)


(b)


(c)

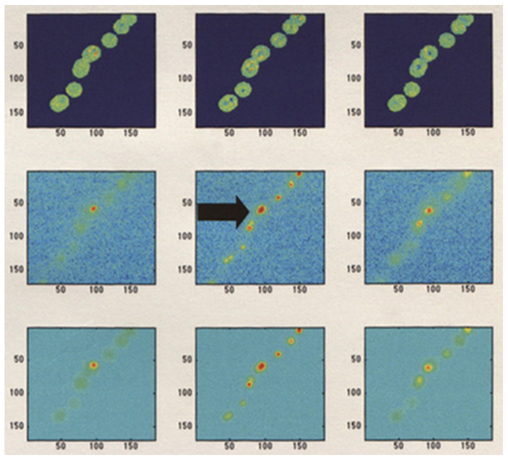

(d)

Figure 26. Here we see results for the Gaussian model with a large disk radius (a), the bead lookup model (b), the nucleus lookup model (c), and the Gaussian model with a small disk radius (b). Each figure has the model image plotted across the bottom row, the raw image data accross the middle row, and the difference between the two plotted across the top row. The spot at which two nuclei are overlapping is indicated by the large black arrow in each image. We can see in the original images that although this spot has a high intensity in the center z-slice, there are also bright spots in the other two z-slices, indicating that two nuclei are present, one which is in focus between the first two z-slices and the other which is in focus between the second two z-slices. All three of the large-disk Gaussian model, bead lookup model, and nucleus lookup model fail to resolve the two overlapping nuclei at two separate z-locations—we can see this by not only observing how dim the model is at this location in z-slices 1 and 3, but also by noting the regions of blue at this location in the difference images for z-slices 1 and 3 (indicating that the model image is of lower intensity than the data images at that location). In the case of the Gaussian model with the small disk, however, the model image appears very similar to the data image, and the difference image has significantly fewer pixels of dark blue and bright red within the disks (indicating that indeed the model is very close to the original images themselves).

other nuclei as well. It is most likely that this run of the Gaussian model was more successful than before due to a larger w_b causing more fitting for noise in the data. The background in these images is very noisy, and the purpose of using a mask when we calculate $J(p)$ is to ensure that we do not waste too much computational effort fitting for the background noise. Looking at a smaller radius around each nucleus (i.e. setting w_b to a lower value) helps ensure that this does not happen; however we cannot make w_b too small or we will no longer be fitting the entirety of each nucleus. In our second run of the Gaussian model w_b is large enough to not risk cutting off crucial information in the nuclei fitting. Thus for good values of w_b, the Gaussian model appears to be successful for fitting nuclei in 3D.

6. Conclusions

By implementing the Metropolis-Hastings algorithm, we have successfully solved a noisy fitting problem using Bayesian inference. In 1D we were able to create toy models that not only succeeded for simple examples but could also handle complex mitosis relationships. When expanding the model to 2D, we successfully fitted real data in two cases, one of which involved a mitosis event. This goes well beyond current tracking software in terms of the ability to handle complicated mitosis and overlap relationships between particles. Although our method failed to settle on the correct solution for one of our real data examples, the model which our method settled around was not too far off from the correct interpretation, and the important aspect of the example (the crossing event) was successfully interpreted.

One unfortunate aspect of our method is the need to choose many constants by hand. We do not have a way to reduce this as they represent information biologists use to interpret image data sets but are highly variable depending on factors such as which microscope is used to collect the images. It may be the case that these constants can be kept the same for most images taken with the same microscope, which would make this problem much less of an issue. Another problem could arise if larger data sets take an unreasonably long number of iterations to settle on the right equilibrium distribution. However, the fact that our 2D mitosis example settled so quickly despite our initial guess being far from the correct interpretation of the data is extremely promising.

We have also shown that we are able to fit the locations of nuclei in 3D using a Gaussian model, provided we only consider a reasonably small area around each nucleus. The incorporation of 3D fitting into our Markov chain Monte Carlo method could help give even better tracking results by further clearing up ambiguities in complicated scenarios such as crossing events.

Appendix A. Pseudocode

A.1. **Proposal Density.** We know from section 2.2 that order to perform Metropolis-Hastings, we need to have a proposal density function $q(x, x')$ in x' centered about x. We have shown that if $q(x, x') = q(x', x)$, the probability of accepting a new state x' from state x depends only on the likelihood of both states, which we can calculate. We therefore need to write a function which, given a state x, will propose a new state x'. We also need this function to choose state x given state x' with equal probability of choosing state x' given state x. We define a random state state function which does this. Each nucleus has the parameters defined in table 1. There is also an extra vector for each nucleus called 'ages' which stores how old the nucleus is at each time frame–age starts at T_{inf} (defined in section 3.2) for nuclei appearing at the beginning or from the edge, and age is 0 after a splitting event, growing by 1 with each time frame. This vector is used in likelihood and not by random state, but it must be updated when we propose a new change in the random state function. The random state function chooses one nucleus to change and a random number between 0 and 1 which will determine which parameter it should change for that nucleus. The specifics for how it determines what change to make starting with parameter vector x can be seen in the pseudocode for the function as follows:

```
Random state($x$)
Let $a$ = [.1 .2 .3 .4 .85 .88 .93 .98] {Probability vector for which parameter to change}
while $x'$ not chosen do
  Let $b$ = random number between 0 and 1
  Let $p$ = random particle
  if $b < a_1$ then
    {Turning a particle on or off}
    if $p$ is on then
      Turn $p$ off
      if $p$ has a parent then
        Update parent's ages
      end if
      Return $x'$
    else
      Turn $p$ on
      if $p$ has a parent then
        Update parent's ages
      end if
      Return $x'$
    end if
  else if $b < a_2$ then
    {Changing a particle's appearance time}
    if $p$ is not on then
      continue
    end if
    Let $d = 1$ or $-1$
    if changing $p$'s current appearance time by $d$ is within $[1, M]$ then
      Change $p$'s appearance time by $d$
      Update $p$'s ages
      if $p$ has a parent then
        Update parent's ages
      end if
      Return $x'$
    else
      continue
    end if
  else if $b < a_3$ then
    {Changing a particle's disappearance time}
    if $p$ is not on then
      continue
    end if
    Let $d = 1$ or $-1$
    if changing $p$'s current disappearance time by $d$ is within $[1, M]$ then
      Change $p$'s disappearance time by $d$
      Return $x'$
    else
      continue
    end if
  else if $b < a_4$ then
    {Changing a particle's parent}
    if $p$ is not on then
      continue
    end if
    Let $r$ be a random particle
    if $r$ isn't on then
      continue
    end if
    if $r$ isn't the same particle as $p$ then
      {Chose a new parent for $p$}
      if $r$'s appearance time $>$ $p$'s appearance time then
        continue
      end if
      Make $r$ be $p$'s parent
      Update $p$'s ages and $r$'s ages
      if $p$ had a different parent $s$ before then
        Update $s$'s ages
      end if
      Return $x'$
    else
      {$r$ we chose is $p$ itself, give $p$ no parent}
      Give $p$ no parent
      Update $p$'s ages
      if $p$ had a parent $s$ before then
        Update $s$'s ages
      end if
      Return $x'$
    end if
  else if $b < a_5$ then
    {Change location of a particle in a single time frame}
    if $p$ is not on then
      continue
    end if
    Let $d = 1$ or $-1$
    Let $t$ = random time frame ($t \in \{1, ..., M\}$)
    if adding $d$ to $p$'s current location at time $t$ is within $[1, M]$ then
      Change $p$'s location at time $t$ by $d$
      **Return** $x'$
    **else**
      Change $p$'s location at time $t$ by $-d$
      **Return** $x'$
    **end if**
  **else if** $b < a_6$ **then**
    {Switch particle identities after a crossing event}
    **if** Can find two particles $r$,$s$ which are within 10 pixels of each other at some single
    time frame $t$ **then**
      Switch identities of $r$ and $s$ from time $t$ onwards
      **if** $r$ and/or $s$ have children **then**
        Update parents of these children
        Update ages of $r$ and $s$ to account for swapping of splitting events
      **end if**
      **Return** $x'$
    **else**
      **continue**
    **end if**
  **else if** $b < a_7$ **then**
    {Change peak intensity of a particle in a single time frame}
    **if** $p$ is not on **then**
      **continue**
    **end if**
    Let $d = 1$ or $-1$
    Let $t$ = random time frame ($t \in \{1, ..., M\}$)
    Change $p$'s peak intensity at time $t$ by $d$
    **Return** $x'$
  **else if** $b < a_8$ **then**
    {Change width of a particle in a single time frame}
    **if** $p$ is not on **then**
      **continue**
    **end if**
    Let $t$ = random time frame ($t \in \{1, ..., M\}$)
    Let $w$ = RANDN() + current width of $p$ {RANDN() returns a pseudorandom value
    drawn from the standard normal distribution}
    **if** $w > 0$ **then**
      Update $p$'s width at time $t$ to $w$
      **Return** $x'$
    **else**
      **continue**
    **end if**
  **else**
    Let $d$ = RANDN()
    Add $d$ to current background intensity
    **Return** $x'$
  **end if**
```

```
end while
```

A.2. **Likelihood Function.** In addition to a proposal density, Metropolis-Hastings also requires that we are able to evaluate a likelihood function for our parameter vector x. For our 2D case we have defined the likelihood in equation (12) with penalties described in section 4. Some of the penalties are accounted for in our random state function—if it chooses certain x' that will have 0 likelihood, it will loop back and choose a new parameter in x to change instead of returning this x'. This does not throw off the reversibility of any change in random state because it is equivalent to setting $L(x') = 0$, which would cause the proposed x' to be rejected in every case. The likelihood function handles the rest of the penalties. The pseudocode for the likelihood function is as follows:

```
Likelihood($x$,$I$)
$M$ = model image based on parameters $x$
$J$ = sum of squared differences between $M$ and $I$ {Defined in equation (2), with $M = I(x)$}
$p_{\text{dist}} = 1$
$p_{\text{appearance}} = 1$
$p_{\text{pd}} = 1$
$p_{\text{v}} = 1$
$p_{\text{young}} = 1$
$p_{\text{pb}} = 1$
$p_{\text{po}} = 1$
if any nucleus $p$ has $w_p < 1$ or $w_p > 6$ then
  $L = 0$
  return $L$
end if
for $i = 1$ to $n_{\text{p}}$ do
  for $j = 1$ to $M - 1$ do
    {Penalty for how far nucleus has travelled}
    $d = \sqrt{(x_{i,j} - x_{i,j+1})^2 + (y_{i,j} - y_{i,j+1})^2}$
    $p_{\text{dist}} = p_{\text{dist}} e^{-\frac{d^2}{2c_{\text{dist}}^2}}$
    if i did not just undergo mitosis then
      {Penalty for change in peak intensity}
      $p_{\text{v}} = p_{\text{v}} e^{-\frac{(v_{i,j} - v_{i,j+1})^2}{2c_{\text{v}}^2}}$
    end if
  end for
  if i is on then
    if i has a parent then
      if i existed from time 1 then
        $p_{\text{po}} = 0$
      end if
      if i's parent is turned off then
        $p_{\text{po}} = 0$
      else if i is born before its parent then
        $p_{\text{pb}} = 0$
      else if i's parent disappeared before i's birth then
```

```
            $p_{\text{pb}} = 0$
          **else**
            {Penalty for being far from parent at birth}
            $d$ = distance of i from it's parent at birth
            $p_{\text{ps}} = p_{\text{ps}} e^{-\frac{d^2}{2c_{\text{dist}}^2}}$
          **end if**
          {Penalty for young nuclei splitting}
          $a$ = age of i's parent at time before i's birth
          $p_{\text{young}} = p_{\text{young}} e^{c_{\text{birth}} \min\left(0, a - T_{\text{inf}}\right)}$
        **else**
          {Penalty for appearing in the middle of the image}
          $d$ = i's distance at appearance time from nearest edge of the image plus outer box of $c_{\text{dist}}$ on each edge
          **if** i appeared after time frame 1 and $d > c_{\text{dist}}$ **then**
            $p_{\text{appearance}} = p_{\text{appearance}} e^{-\frac{d^2}{2c_{\text{dist}}^2}}$
          **end if**
        **end if**
        {Penalty for appearing in the middle of the image}
        $d$ = i's distance at disappearance time from nearest edge of the image plus outer box of $c_{\text{dist}}$ on each edge
        **if** i disappeared before time frame M and $d > c_{\text{dist}}$ **then**
          $p_{\text{appearance}} = p_{\text{appearance}} e^{-\frac{d^2}{2c_{\text{dist}}^2}}$
        **end if**
      **end if**
    **end for**
    $L = p_{\text{dist}} p_{\text{appearance}} p_{\text{pd}} p_{\text{v}} p_{\text{young}} p_{\text{pb}} p_{\text{po}} e^{-J/T}$ {Calculate likelihood from equation (12)}
    **return** $L$
```

References

[1] A. S. Gladfelter, A. K. Hungerbuehler, and P. Philippsen. *Asynchronous mitoses in multinucleated cells.* J. Cell Biol. **172** (2006), 347–362.

[2] J. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems.* New York: Springer, 2004.

[3] D. J. C. MacKay, "Introduction to Monte Carlo Methods," in *Learning in Graphical Models*, M. I. Jordan, editor. Cambridge, MA: MIT Press, 1999.

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

[5] E. Tice. *Automated Tracking of Dividing Nuclei in Microscopy Videos of Living Cells.* Undergraduate thesis, Dartmouth College, June 2009.