





Article

# Stochastic Neural Networks for Automatic Cell Tracking in Microscopy Image Sequences of Bacterial Colonies

Sorena Sarmadi<sup>1</sup>, James J. Winkle<sup>1</sup>, Razan N. Alnahhas<sup>2</sup> , Matthew R. Bennett<sup>3</sup> , Krešimir Josić<sup>1,4</sup> ,  
Andreas Mang<sup>1</sup>  and Robert Azencott<sup>1,\*</sup>

<sup>1</sup> Department of Mathematics, University of Houston, Houston, TX 77204, USA; sorena.sarmadi@gmail.com (S.S.); wink@utexas.edu (J.J.W.); kresimir.josic@gmail.com (K.J.); andreas@math.uh.edu (A.M.)

<sup>2</sup> Department of Biomedical Engineering, Boston University, Boston, MA 02215, USA; rnalnahhas@gmail.com

<sup>3</sup> Departments of Biosciences and Bioengineering, Rice University, Houston, TX 77005, USA; matthew.bennett@rice.edu

<sup>4</sup> Department of Biology and Biochemistry, University of Houston, Houston, TX 77204, USA

\* Correspondence: razencot@math.uh.edu; Tel.: +1-713-743-3489

**Abstract:** Our work targets automated analysis to quantify the growth dynamics of a population of bacilliform bacteria. We propose an innovative approach to frame-sequence tracking of deformable-cell motion by the automated minimization of a new, specific cost functional. This minimization is implemented by dedicated Boltzmann machines (stochastic recurrent neural networks). Automated detection of cell divisions is handled similarly by successive minimizations of two cost functions, alternating the identification of children pairs and parent identification. We validate the proposed automatic cell tracking algorithm using (i) recordings of simulated cell colonies that closely mimic the growth dynamics of *E. coli* in microfluidic traps and (ii) real data. On a batch of 1100 simulated image frames, cell registration accuracies per frame ranged from 94.5% to 100%, with a high average. Our initial tests using experimental image sequences (i.e., real data) of *E. coli* colonies also yield convincing results, with a registration accuracy ranging from 90% to 100%.

**Keywords:** stochastic neural networks; cell tracking; microscopy image analysis; detection-and-association methods

**MSC:** 62H35; 62M45



**Citation:** Sarmadi, S.; Winkle, J.J.; Alnahhas, R.N.; Bennett, M.R.; Josić, K.; Mang, A.; Azencott, R. Stochastic Neural Networks for Automatic Cell Tracking in Microscopy Image Sequences of Bacterial Colonies. *Math. Comput. Appl.* **2022**, *27*, 22. <https://doi.org/10.3390/mca27020022>

Academic Editor: Leonardo Trujillo

Received: 16 December 2021

Accepted: 23 February 2022

Published: 2 March 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

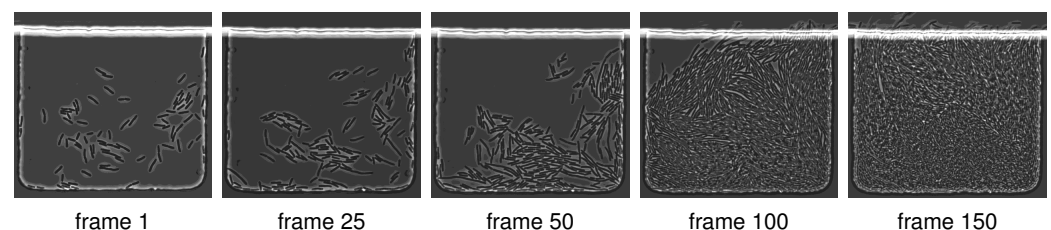
## 1. Introduction

Technology advances have led to increasing magnitudes of data generation with increasing levels of precision [1,2]. However, data generation presently far outpaces data analysis and drives the requirement for analyzing such large-scale data sets with automated tools [3–5]. The main goal of the present work is to develop computational methods for an automated analysis of microscopy image sequences of colonies of *E. coli* growing in a single layer. Such recordings can be obtained from colonies growing in microfluidic devices, and they provide a detailed view of individual cell-growth dynamics as well as population-level, inter-cellular mechanical and chemical interactions [6–8].

However, to understand both variability and lineage-based correlations in cellular response to environmental factors and signals from other cells requires the tracking of large numbers of individual cells across many generations. This can be challenging, as large cell numbers tightly packed in microfluidic devices can compromise spatial resolution, and toxicity effects can place limits on the temporal resolution of the recordings [9,10]. One approach to better understand and control the behavior of these bacterial colonies is to develop computational methods that capture the dynamics of gene networks within single cells [6,11,12]. For these methods to have a practical impact, one ultimately has to fit the

models to the data, which allows us to infer hidden parameters (i.e., characteristics of the behavior of cells that cannot be measured directly). Image analysis and pattern recognition techniques for biological imaging data [13–15], like the methods developed in the present work, can be used to track lineages and thus automatically infer how gene expression varies over time. These methods can serve as an indispensable tool to extract information to fit and validate both coarse and detailed models of bacterial population, thus allowing us to infer model parameters from recordings.

Here, we describe an algorithm that provides *quantitative* information about the population dynamics, including the life cycle and lineage of cells within a population from recordings of cells growing in a mono-layer. A typical sequence of frames of cells growing in a microfluidic trap is shown in Figure 1. We describe the design and validation of algorithms for tracking individual cells in sequences of such images [11,16,17]. After segmentation of individual image frames to identify each cell, tracking individual cells from frame to frame is a combinatorial problem. To solve this problem we take into account the unknown cell growth, cell motion, and cell divisions that occur between frames. Segmentation and tracking are complicated by imaging noise and artifacts, overlap of bacteria, similarity of important cell characteristics across the population (shape; length; and diameter), tight packing of bacteria, and large interframe durations resulting in significant cell motion, and up to a 30% increase in individual cell volume.



**Figure 1.** Typical microscopy image sequence. We show five frames out of a total of 150 frames of an image sequence showing the dynamics of *E. coli* in a microfluidic device [18] (real laboratory image data). These cells are about  $1\ \mu\text{m}$  in diameter and on average  $3\ \mu\text{m}$  in length, and they divide about every 30 min. The original images exported from the microscope are  $0.11\ \mu\text{m}/\text{pixel}$ . We report results for these real datasets in Section 4.

### 1.1. Related Work

The present work focuses on tracking *E. coli* in a time series of images. A comparison of different cell-tracking algorithms can be found in [19,20]. Multi-object tracking in video sequences and object recognition in time series of images is a challenging task that arises in numerous applications in computer vision [21,22]. In (biomedical) image processing, motion tracking is often referred to as “*image registration*” [19,23–26] or “*optical flow*” [27–30]. Typical solutions used in the defense industry, for instance, track *small numbers* of fast moving targets by image sequence analysis at pixel levels and use sophisticated reconstruction of the optical flow, combined with real time segmentation, and quick combinatoric exploration at each image frame. Initially, we did implement several well-known algorithms for reconstruction of the optical flow, but the results we obtained were not satisfactory due to long interframe times and high noise levels. Moreover, we are not interested in tracking individual pixels but rather cells (i.e., rod-shaped, deformable shapes), while recognizing events of cell division and recording cell lineage. Consequently, we decided to first segment each image frame to isolate each cell, and then to match cells between successive frames.

As for the problem at hand, one approach proposed in prior work to simplify the tracking task is to make the experimental setup more rigid by confining individual cell lineages to small tubes; the associated microfluidic device is called a “*mother machine*” [31–36]. The microfluidic devices we consider here yield more complicated data as cells are allowed to move and multiply freely in two dimensions (constrained to a mono-layer). We refer to Figure 1 for a typical sequence of experimental images considered in the present work.

Turning to methods that work on more complex biological cell imaging data, we can distinguish different classes of tracking methods. “*Model-based evolution methods*” operate on the image intensities directly. They rely on particle filters [37–39] or active contour models [40–44]. These methods work well if the cells are not tightly packed. However, they may lead to erroneous results if the cells are close together, the inter-cellular boundaries are blurry, or the cells move significantly. Our work belongs to another class—the so called “*detection-and-association methods*” [45–55], which first detect cells in each frame and then solve the tracking problem/association task across successive frames. (We note that the segmentation and tracking of cells does not necessarily need to be implemented in two distinct steps. In many image sequence analyses, implementing these two steps jointly can be beneficial [37,49,54,56–58]. However, for the clarity of exposition and easier implementation of our new tracking technique, we present these steps separately.) Doing so necessitates the segmentation of cells within individual frames. We refer to [59] for an overview of cell segmentation approaches. Deep learning strategies have been widely used for this task [5,50,54,58,60–65]. We consider a framework based on convolutional neural networks (CNNs). Others have also used CNNs for cell segmentation [62,64,66–68]. We omit a detailed discussion of our segmentation approach within the main body of this paper, as we do not view it as our main contribution (see Section 1.2). However, the interested reader is referred to Appendix D for some insights. To solve the tracking problem after the cell detection, many of the methods cited above use hand-crafted association scores based on the proximity of the cells and shape similarity measures [46,48,51,54]. We follow this approach here. We note that we not only consider local association scores between cells but also include measures for the integrity of a cell’s neighborhood (i.e., “*context information*”).

Our method is tailored for tracking cells in tightly packed colonies of rod-shaped *E. coli* bacteria. This problem has been considered previously [5,45,49,52]. However, we are not aware of any large-scale datasets that provide ground truth tracking data for these types of recordings, but note that there are community efforts for providing a framework for testing cell tracking algorithms [20,69] (see, e.g., [37,70]). (Cell tracking challenge: <http://celltrackingchallenge.net> (accessed on 15 December 2021).) Works that consider these data are for example [37,54,57,58,62,67]. The cells in this dataset have significantly different characteristics compared to those considered in the present work. As we describe below, our approach is based on distinct characteristics of the bacteria cells and, consequently, does not directly apply to these data. Therefore, we have developed our own validation and calibration framework (see Section 2.1).

Standard graph matching algorithms (see, e.g., [71]) do not directly apply to our problem. Indeed, a fundamental complication is that cells can divide between successive images. Hence, each assignment from one frame to its successor is not a one-to-one mapping but a one-to-many correspondence. More advanced graph matching strategies are described in [72,73]. Graph-based matching strategies for cell-tracking that are somewhat related to our approach are described in [70,74–77]. Like the methods mentioned above, they consider various association scores for tracking. Individual cells are represented as nodes, and neighbors are connected through edges. Our approach also introduces cost terms for structural matching of local neighborhoods by specific scoring for single nodes, pairs of nodes, and triplets of nodes, after a (modified) Delaunay triangulation. By using a graph-like structure, cell divisions can be identified by detecting changes in the topology of the graph [75,76]. We tested a similar strategy, but came to the conclusion that we cannot reliably construct neighborhood networks between frames for which topology changes only occur due to cell division; the main issue we observed is that the significant motion of cells between frames can introduce additional topology changes in our neighborhood structure. Consequently, we decided to relax these assumptions.

Refs. [71,78–80] implement multi-target tracking in videos by stochastic models based on random finite set densities and variants thereof. The fit to the data are based on Gibbs sampling to maximize the posterior likelihood. A key challenge of these approaches is the estimation of an adequate finite number of Gibbs sampling iterations when one computes

posterior distributions. Most Gibbs samplers are ergodic Markov chains on a finite but huge state spaces, so that their natural exponential rate of convergence is not a practically reassuring feature.

As mentioned above, some recent works jointly solve the tracking and segmentation problem [37,49,54,56–58]. Contrary to observations we have made in our data, these approaches rely (with the exception of [49]) on the fact that the tracking problem is inherent to the segmentation problem (“*tracking-by-detection methods*” [54]; see also [5]). That is, the key assumption made by many of these algorithms is that cells belonging to the same lineage overlap across frames (see also [47]). In this case, cell-overlap can serve as a good proxy for cell-tracking [54]. We note that in our data we cannot guarantee that the frame rate is sufficiently high for this assumption to hold. Refs. [56,57,67] exploited machine learning techniques for segmentation *and* motion tracking. One key challenge here is to provide adequate training data for these methods to be successful. Here, we describe simulation-based techniques that can be extended to produce training data, which we use for parameter tuning [12,81].

The works that are most similar to ours are [45,49,52]. They perform a local search to identify the best cell-tracking candidates across frames. One key difference across these works are the matching criteria. Moreover, Refs. [45,49] employ a local greedy-search, whereas we consider stochastic neural network dynamics for optimization. Ref. [52] constructs score matrices within a score based neighborhood tracking method; an integer programming method is used to generate frame-to-frame correspondences between cells and the lineage map. Other approaches that consider linear programming to maximize an association score function for cell tracking can be found in [47,54,82].

As we have mentioned in the abstract, we obtain a tracking accuracy that ranges from 90% to 100%, respectively. Overall, our method is competitive with existing approaches: For example, Ref. [45] reports a tracking accuracy of up to 97% for data that are similar to ours, while Ref. [74] reports a tracking accuracy (spatial, temporal, and cell division detection) at the order of 95% (between about 93% and 98%, respectively). The second group also reports results for their prior approach [75], with an accuracy at the order of 90% (ranging from about 87% to 92%, respectively). Accuracies reported in [77] range from about 92% to 97%, respectively. This work also includes a comparison to one of their earlier approaches [76] with an accuracy of up to 85% and 89% if the datasets are pre-aligned. We note that the data considered in [74–77] are quite different from ours. Refs. [37,54,57,58,70] consider the data from the cell tracking challenge [20,69] to evaluate the performance of their methods. As in the previously mentioned work, these data are again quite different from ours. To evaluate the performance of the methodology, the so-called *acyclic oriented graph matching measure* [83] is considered. We refer to the webpage of the cell tracking challenge for details on the evaluation metrics (see <http://celltrackingchallenge.net/evaluation-methodology>, accessed on 15 December 2021). Based on these, the reported tracking scores are between 0.873 and 0.902 [37], 0.901 and 1.00 [70], 0.950 and 0.987 [54], 0.788 and 0.982 [58] and 0.765 and 0.915 [57] depending on the considered data set, respectively.

### 1.2. Contributions

For image segmentation, we first apply two well-known, powerful variational segmentation algorithms to generate a large training set of correctly delineated single cells. We can then train a CNN dedicated to segmenting out each single cell. Using a CNN significantly reduces the runtime of our computational framework for cell identification. The frame-to-frame tracking of individual cells in tightly packed colonies is a significantly more challenging task, and is hence the main topic discussed in the present work. We develop a set of innovative automatic cell tracking algorithms based on the successive minimization of three dedicated cost functionals. For each pair of successive image frames, minimizing these cost functionals over all potential cell registration mappings poses significant computational and mathematical challenges. Standard gradient descent algorithms are inefficient for these discrete and highly combinatorial minimization problems. Instead, we implement



the stochastic neural network dynamics of BMs, with architectures and energy functions tailored to effectively solve our combinatorial tracking problem. Our major contributions are: (i) The design of a multi-stage cell tracking algorithm that starts with a parent–children pairing step, followed by removal of identified parent–children triplets, and concludes with a cell-to-cell registration step. (ii) The design of dedicated BM architectures, with several energy functions, respectively, minimized by true parent–children pairing and by true cell-to-cell registration. Energy minimizations are then implemented by simulation of BM stochastic dynamics. (iii) The development of automatic algorithms for the estimation of unknown weight parameters of our BM energy functions, using convex-concave programming tools [84–86]. (iv) The evaluation of our methodology on synthetic and real image sequences of cell colonies. The massive effort involved in human expert annotation of cell colony recordings limits the availability of “ground truth tracking” data for dense bacterial colonies. Therefore, we first validated the accuracy of our cell tracking algorithms on recordings of simulated cell colonies, generated by the dedicated cell colony simulation software [12,81]. This provided us with ground truth frame-by-frame registration for cell lineages, enabling us to validate our methodology.

### 1.3. Outline

In Section 2, we describe the synthetic image sequence (see Section 2.1) and experimental data (see Section 2.2) of cell colonies considered as benchmarks for our cell tracking algorithms. In Section 2.3, we describe key cell characteristics considered in our tracking methodology to define metrics that enter our cost functionals. Our tracking approach is developed in greater detail in Section 3. We define valid cell registration mappings between successive image frames in Section 3.1. We outline how to automatically calibrate the weights of our various penalty terms in Section 3.2. Our algorithms for pairing parent cells with their children and for cell-to-cell registration are developed in Sections 3.3–3.9. We present our main validation results on long image sequences (time series of images) in Section 4 and conclude with Section 5.

## 2. Datasets

Below, we introduce the datasets used to evaluate the performance of the proposed methodology. The synthetic data are described in Section 2.1. The experimental data (real imaging data) are described in Section 2.2.

### 2.1. Synthetic Videos of Simulated Cell Colonies

To validate our cell tracking algorithms, we consider simulated image sequences of dense cell populations. We refer to [12,81] for a detailed description of this mathematical model and its implementation. (The code for generating the synthetic data has been released at <https://github.com/jwinkle/eQ>, accessed on 15 December 2021) The simulated cell colony dynamics are driven by an agent based model [12,81], which emulates live colonies of growing, moving, and dividing rod-like *E. coli* cells in a 2D microfluidic trap environment. Between two successive frames  $J, J_+$ , cells are allowed to move until they nearly bump into each other, and to grow at multiplicative rate denoted  $g.rate$  with an average value of 1.05 per minute.

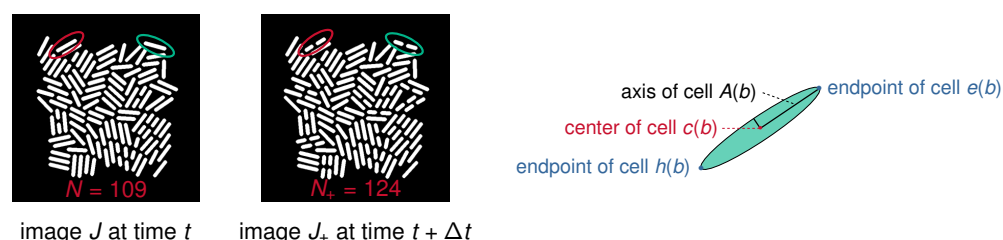
The cells are modeled as 2D spherocylinders of constant  $1\ \mu\text{m}$  width. Each cell grew exponentially in length with a doubling time of 20 min. To prevent division synchronization across the population when a mother cell of length  $L_{\text{div}}$  divides, the two daughter cells are assigned random birth lengths  $L_0(b_1) = L_1 = \delta L_{\text{div}}$  and  $L_0(b_2) = L_2 = (1 - \delta)L_{\text{div}}$ , where  $\delta > 0$  is a random number sampled independently at each division from a uniform distribution on  $[0.45, 0.55]$ . Consequently, a bacterial cell  $b$  of length  $L_{\text{div}}$  divides into two cells  $b_1$  and  $b_2$ , their lengths  $L_1, L_2$  satisfy  $L_1 + L_2 = L_{\text{div}}$  and  $L_i/L_{\text{div}}, i = 1, 2$ , is a random number. The cells have a length of approximately  $2\ \mu\text{m}$  after division and  $4\ \mu\text{m}$  right before division. We refer to [81] for additional details. The simulation keeps track of cell lineage, cell size, and cell location (among other parameters). The main output of each

such simulation considered here is a binary image sequence of the cell colony with a fixed interframe duration. Each such synthetic image sequence is used as the sole input to our cell tracking algorithm. The remaining meta-data generated by the simulations are only used as ground truth to evaluate the performance of our tracking algorithms.

We consider several benchmark datasets of *synthetic* image sequences of simulated cell colonies of different complexity. We refer to these benchmarks as BENCH1 (500 frames), BENCH2 (300 frames), BENCH3 (300 frames), and BENCH6 (100 frames), with an interframe duration of 1, 2, 3 and 6 min, respectively. Notice that there is no explicit noise on the growth rate. However, due to the crowding of cells, the growth rate will vary from cell-to-cell. The generated binary images are of size  $600 \times 600$  pixels. We summarize these benchmarks in Table 1. The associated image sequences involve between 100 up to 500 frames, respectively. In Figure 2, we display an example of two simulated consecutive frames separated by 1 min. To simplify our presentation and validation tests, we control our simulations to make sure that cells will not exit the region of interest from one frame to the next, and we exclude cells that are only partially visible in the current frames.

**Table 1.** Benchmark datasets. To test the tracking software, we consider simulated data. We have generated data of varying complexity with different interframe durations. We note that we also consider these data to train our algorithms for tracking cells. We report the label for each dataset, the interframe duration, as well as the number of frames generated. We set the *cell growth factor* to  $g.rate = 1.05$  per min. We refer to the text for details about how these data have been generated.

Label	Interframe Duration	Number of Frames
BENCH1	1 min	500
BENCH2	2 min	300
BENCH3	3 min	300
BENCH6	6 min	100



**Figure 2.** Simulated data and cell characteristics considered in the proposed algorithm. **(Left):** Two successive images generated by dynamic simulation for a colony of rod-shaped bacteria. The left image  $J$  displays  $N = 109$  cells at time  $t$ . At time  $t + \Delta t$  with  $\Delta t = 1$  min, cells have moved and grown, and some have divided. These cells are displayed in image  $J_+$ , which contains  $N_+ = 124$  cells. We highlight two cells that have undergone a division between the frames (red and green ellipses). **(Right):** Geometry of a rod shaped bacterium. We consider different quantities of interest in the proposed algorithm. These include the center  $c(b)$  of a cell, the two end points  $e(b)$  and  $h(b)$ , and the long axis  $A(b)$ , respectively.

## 2.2. Laboratory Image Sequences (Real Biological Data)

We also verify the performance of our approach on real datasets of *E. coli* bacteria. These bacteria are about  $1 \mu\text{m}$  in diameter and on average  $3 \mu\text{m}$  in length, and they divide about every 30 min. The original images exported from the microscope are  $0.11 \mu\text{m}/\text{pixel}$ . The microscopy experimental data were obtained using JS006 [87] (BW25113  $\Delta araC \Delta lacI$ ) *E. coli* strains containing a plasmid constitutively expressing yellow or cyan fluorescent protein (*sfyfp* or *sfcfp*) for identification. The plasmid also contains an ampicillin resistance gene and p15A origin. These cells were grown overnight in LB medium with  $100 \mu\text{g}/\text{mL}$  ampicillin for 18 h. These cultures were diluted in the morning into 1/1000 into 50 mL fresh LB with  $100 \mu\text{g}/\text{mL}$  ampicillin and grown for 3 h until they reached an OD600 of about

0.3. The cells were then concentrated by centrifuging 30 mL of culture at  $2000\times g$  for 5 min and then resuspending in 10 mL of fresh LB. The concentrated culture was loaded into a hallway microfluidic device prewarmed and flushed with 0.1% (v/v) Tween-20 [88]. In the microfluidic device, the cells were provided with continuous fresh LB with 100  $\mu\text{g/mL}$  ampicillin and 0.075% (v/v) Tween-20. The microfluidic device was placed onto an  $60\times$  oil objective and imaged every 6 min at phase contrast, YFP, and CFP filter settings using an inverted fluorescence microscope. We show a representative dataset in Figure 1.

### 2.3. Cell Characteristics

Next, we discuss characteristics of the *E. coli* bacteria important for our tracking algorithm.

**Cell Geometry.** In accordance with the dynamics of bacterial colonies in microfluidic traps, the dynamic simulation software generates colonies of rod-shaped bacteria. Cell shapes can be approximated by long and thin ellipsoids, which are geometrically well identified by their center, their long axis, and the two endpoints of this long axis. The center  $c(b)$  is the centroid of all pixels belonging to cell  $b$ . The long axis  $A(b)$  of cell  $b$  is computed by principal component analysis (PCA). The endpoints  $e(b)$  and  $h(b)$  of cell  $b$  are the first and last cell pixels closest to  $A(b)$ ; see Figure 2 (right) for a schematic illustration.

**Cell Neighbors.** For each image frame  $J$ , denote  $B = B(J)$  as the set of fully visible cells in  $J$ , and by  $N = N(J) = \text{card}(B)$  the number of these cells. Let  $V$  be the set of all cell centers  $c(b)$  with  $b \in B$ . Denote  $\text{del}V$  the Delaunay triangulation [89] of the finite planar set  $V$  with  $N$  vertices. We say that two cells  $b_1, b_2$  in  $B$  are *neighbors* if they verify the following three conditions: (i)  $(b_1, b_2)$  are connected by the edge  $\text{edg}$  of one triangle in  $\text{del}V$ . (ii) The edge  $\text{edg}$  does not intersect any other cell in  $B$ . (iii) Their centers verify  $\|c(b_1) - c(b_2)\| \leq \rho$ , where  $\rho > 0$  is a user defined parameter.

For the synthetic images of size  $600 \times 600$  that we considered (see Section 2.1), we take  $\rho = 80$  pixels. We write  $b_1 \sim b_2$  for short, whenever  $b_1, b_2$  are neighbors (i.e., satisfy the three conditions identified above).

**Cell Motion.** Let  $J, J_+$  denote two successive images (i.e., frames). Denote  $B = B(J)$ ,  $B_+ = B(J_+)$  as the associated sets of cells. Superpose temporarily the images  $J$  and  $J_+$  so that they then have the same center pixel. Any cell  $b \in B$ , which does not divide in the interframe  $J \rightarrow J_+$ , becomes a cell  $b_+ \in B_+$  in image  $J_+$ . The “motion vector” of cell  $b$  from frame  $J$  to  $J_+$  is then defined by  $v(b) = c(b_+) - c(b)$ . If the cell  $b$  does divide between  $J$  and  $J_+$ , denote  $b_{\text{div}}$  as the last position reached by cell  $b$  at the time of cell division, and define similarly the motion  $v(b) = c(b_{\text{div}}) - c(b)$ . In our experimental recordings of real bacterial colonies with interframe duration 6 min, there is a *fixed number*  $w > 0$  such that  $\|v(b)\| \leq w/2$  for all cells  $b \in B(J)$  for all pairs  $J, J_+$ . In particular, we observed that, for real image sequences,  $w = 100$  pixels is an adequate choice. Consequently, we select  $w = 100$  pixels for all simulated image sequences of BENCH6. For BENCH1, we select  $w = 45$  pixels, again based on a comparison with real experimental recordings. Overall, the meta-parameter  $w$  is assumed to be a fixed number and to be known, since  $w/2$  is an observable upper bound for the cell motion norm for a particular image sequence of a lab experiment.

**Target Window.** Recall that  $J, J_+$  are temporarily superposed. Let  $U(b) \subset J_+$  be a square window of width  $w$ , with the same center as cell  $b$ . The *target window*  $W(b)$  is the set of all cells in  $B_+$  having their centers in  $U(b)$ . Since  $\|v(b)\| \leq w/2$ , the cell  $b_+$  must belong to the target window  $W(b) \subset B_+$ .

## 3. Methodology

### 3.1. Registration Mappings

Next, we discuss our assumptions on a valid registration mapping that establishes cell-to-cell correspondences between two frames. Let  $J, J_+$  denote two successive images, with cell sets  $B$  and  $B_+$ , respectively. As above, we let  $N = \text{card}(B)$ , and  $N_+ = \text{card}(B_+)$ . Our goal is to track each cell from  $J$  to  $J_+$ . For each cell  $b \in B$ , there exist three possible evolutions between  $J$  and  $J_+$ :

**Case 1:** Cell  $b \in B$  did not divide in the interframe  $J \rightarrow J_+$ , and has become a cell  $f(b) \in B_+$ ; that is,  $f(b)$  has grown and moved during the interframe time interval.

**Case 2:** Cell  $b \in B$  divided between  $J$  and  $J_+$ , and generated two children cells  $b_1, b_2 \in B_+$ ; we then denote  $f(b) = (b_1, b_2) \in B_+ \times B_+$ .

**Case 3:** Cell  $b \in B$  disappeared in the interframe  $J \rightarrow J_+$ , so that  $f(b)$  is not defined.

To simplify our exposition, we *ignore Case 3*. We discuss Case 3 in greater detail in the conclusions in Section 5. Consequently, a valid (true) registration mapping  $f$  will take values in the set  $\{B_+\} \cup \{B_+ \times B_+\}$ .

### 3.2. Calibration of Cost Function Weights

With the notation we introduced, fix any two finite sets  $A, A_+$ . Let  $G := \{g : A \rightarrow A_+\}$  be the set of all mappings  $g : A \rightarrow A_+$ . Fix  $m$  penalty functions  $\text{pen}_k(g) \geq 0$ ,  $k = 1, \dots, m$ . Let  $g^* \in G$  be the ground truth mapping we want to discover through minimization in  $g$  of some given cost function  $\text{COST}(g)$  defined by the linear combination of the penalty functions  $\text{pen}_k(g)$ , the contributions of which are controlled by the cost function weights  $\lambda_k > 0$ . In this section, we present a generic *weight calibration algorithm*, extending a technique introduced and applied in [90,91] for Markov random fields based image analysis.

The cost function must perform well (with the same weights) for hundreds of pairs of (synthetic) images  $J, J^+$ . We consider one such synthetic pair for which the ground truth registration mapping  $f \in G$  is known, and use it to compute an adequate set of weights, which will then be used on all other synthetic pairs  $J, J^+$ . Notice that, for experimental recordings of real cell colonies, no ground truth registration mappings  $f$  are available. In this case,  $f$  should be replaced by a set of user constructed, correct *partial* mappings defined on small subsets of  $A$ . The proposed weight calibration algorithm will also work in those situations.

We now show how knowing one ground truth mapping  $f$  can be used to derive the best feasible weights ensuring that  $f$  should be a plausible minimizer of the cost functional  $\text{COST}(g)$  over  $g \in G$ . Let  $\text{PEN}(g) = [\text{pen}_1(g), \dots, \text{pen}_m(g)]$  be the vector of  $m$  penalties for any mapping  $g \in G$ . Let  $\Lambda = [\lambda_1, \dots, \lambda_m]$  be the weight vectors. Then,  $\text{COST}(g) = \langle \Lambda, \text{PEN}(g) \rangle$ . Replacing  $g$  by another mapping  $h \neq g$  induces the penalty changes  $\Delta \text{PEN}_{g,h} = \text{PEN}(h) - \text{PEN}(g)$  and the cost change  $\Delta \text{COST}(g, h) = \langle \Lambda, \Delta \text{PEN}_{g,h} \rangle$ . Now, fix any known ground truth mapping  $f \in G$ . We want  $f$  to be a minimizer of  $\text{COST}$ , so we should have  $\Delta \text{COST}(f, f') \geq 0$  for all modifications  $f \rightarrow f' \in G$ .

For each  $a \in A$ , select an arbitrary  $s(a) \in W(a)$  (where  $W(a)$  is the target window for cell  $a$ ; see Section 2.3), to define a new mapping  $f' = f'_a$  from  $A$  to  $A_+$  by  $f'_a(a) = s(a)$ , and  $f'_a(x) \equiv f(x)$  for all  $x \neq a$ . Since  $f$  is a minimizer of  $\text{COST}$ , this single point modification  $f \rightarrow f'_a$  must generate the following cost increase

$$\langle \Lambda, \Delta \text{PEN}(f, f'_a) \rangle = \Delta \text{COST}(f, f'_a) \geq 0.$$

Denote  $V_a \in \mathbb{R}^m$  the vector  $V_a = \Delta \text{PEN}(f, f'_a)$ . Then, the positive vector  $\Lambda \in \mathbb{R}^m$ ,  $\Lambda \succeq 0$ , should verify the set of linear constraints  $\langle \Lambda, V_a \rangle \geq 0$  for all  $a \in A$ . There may be too many such linear constraints. Consequently, we *relax* these constraints by introducing a vector  $y = [y(a)] \in \mathbb{R}^{\text{card}(A)}$ ,  $y \succeq 0$ , of slack variables  $y(a) \geq 0$  indexed by all the  $a \in A$ . (In optimization, slack variables are introduced as additional unknowns to transform inequality constraints to an equality constraint and a non-negativity constraint on the slack variables.) We require the unknown positive vector  $\Lambda$  and the slack variables vector  $y$  to verify the system of linear constraints:

$$\begin{aligned} \langle \Lambda, V_a \rangle + y(a) &= 0 & \text{for all } a \in A \\ \Lambda &\succeq 0, \quad y \succeq 0 \\ \langle \Lambda, Z \rangle &\leq 1000 \end{aligned} \tag{1}$$



where  $Z = [1, \dots, 1] \in \mathbb{R}^m$ . The normalizing constant 1000 can be arbitrarily changed by rescaling. We seek high positive values for  $\Delta \text{COST}(f, f'_a)$  and small  $L_1$ -norm for the slack variable vector  $y$ . Thus, we will seek two vectors  $\Lambda \in \mathbb{R}^m$  and  $y \in \mathbb{R}^{\text{card}(A)}$  solving the following *convex-concave* minimization problem, where  $\gamma > 0$  is a user selected (large) meta parameter:

$$\underset{\Lambda, y}{\text{minimize}} \quad \gamma \|y\|_{L1} - \sum_{a \in A} [\langle \Lambda, V_a \rangle]^+ \quad (2)$$

subject to (1), where we denote  $[x]^+ := \max(x, 0)$  for arbitrary  $x$ . To numerically solve the constrained minimization problem (2), we use the libraries CVXPY and DCCP (disciplined convex-concave programming) [84–86]. DCCP is a package for convex-concave programming designed to solve non-convex problems. (DCCP can be downloaded at <https://github.com/cvxgrp/dccp> (last accessed on 20 January 2022)) It can handle objective functions and constraints with any known curvature as defined by the rules of disciplined convex programming [92]. We give examples of numerically computed weight vectors  $\Lambda$  below. The computing time was less than 30 s for the data that we have prepared. For simplicity, we just considered one step changes in our computations, which make the overlap penalty weak. To increase the accuracy of the model, it is possible to consider a larger number of samples (i.e., multi-step changes). Note that the solutions  $\Lambda$  of (2) are of course not unique, even after normalization by rescaling.

### 3.3. Cell Divisions and Parent–Children Short Lineages

Next, we discuss how we tackle the assignment problem when cells divide.

#### 3.3.1. Cell Divisions

We now outline a cost function based methodology to detect cell divisions. The first step will be to seek the most likely parent for each potential pair of children cells. Fix two successive synthetic image frames  $J, J_+$  with short interframe time equal to 1 minute. Their cell sets  $B, B_+$  have cardinality  $N$  and  $N_+$ , respectively. For our synthetic image sequences, all cells  $b \in B$  still exist in  $B_+$ —either as whole cells or after dividing into two children cells, and no new cell enters the field of view during the interframe  $J \rightarrow J_+$ . This forces  $N_+ \geq N$ , and implies that the number  $DIV$  of cell divisions occurring in this interframe verifies  $DIV = N_+ - N$ . Each children pair  $(b_1, b_2) \in B_+ \times B_+$  is born from a single parent  $b \in B$ . Thus, the set *trueCH* of all such *true children pairs* must then verify

$$\text{card}(\text{trueCH}) = DIV = N_+ - N. \quad (3)$$

For our video recordings of actual cell populations, during any interframe, we may have  $n_{out}$  cells exiting the field of view and  $n_{in}$  cells entering it, so that  $|\text{card}(\text{trueCH}) - DIV|$  may be of the order of  $n_{in} + n_{out}$ . To take this into account, we *relax* the constraint in (3) as follows:

$$|\text{card}(\text{trueCH}) - DIV| \leq REL, \quad (4)$$

where  $REL$  is a fixed bound estimated from our experiments. For simplicity, we have restricted our methodology to the situation where  $n_{in}$  and  $n_{out}$  are always 0. However, even in that case, there was a computational advantage to using the slightly relaxed constraint (4) with  $REL = 1$ .

#### 3.3.2. Most Likely Parent Cell for a Given Children Pair

For successive images  $J, J_+$  with 1 min interframe, define the set *PCH* of *plausible children pairs* by

$$PCH = \{(b_1, b_2) \in B_+ \times B_+ \text{ with centers } c_1, c_2 \text{ verifying } \|c_1 - c_2\| < \tau\}, \quad (5)$$

where the threshold  $\tau > 0$  is user selected and fixed for the whole benchmark set BENCH1 of synthetic image sequences.

To evaluate if a pair of cells  $(b_1, b_2) \in PCH$  can qualify as a pair of children generated by division of a parent cell  $b \in B$ , we now quantify the “geometric distortion” between  $b$  and  $(b_1, b_2)$ . Cell division of  $b$  into  $b_1, b_2 \in B_+$  occurs with small motions of  $b_1, b_2$ . During the short interframe duration, the initial centers  $c_1, c_2$  of  $b_1, b_2$  in image  $J$  move by at most  $w/2$  pixels each (see Section 2.3), and their initial distance to the center  $c$  of  $b$  is roughly at most  $\|A(b)\|/4$ , where  $A(b)$  is the long axis of cell  $b$ . Hence, the centers  $c, c_1, c_2$  of  $b, b_1, b_2$  should verify the constraint

$$\max\{\|c_1 - c\|, \|c_2 - c\|\} \leq w + \|A\|/4. \quad (6)$$

Define the set *SHLIN* of potential *short lineages* as the set all triplets  $(b, b_1, b_2)$  with  $b \in B, (b_1, b_2) \in PCH$ , verifying the preceding constraint (6). For each potential lineage  $(b, b_1, b_2) \in SHLIN$ , define three terms penalizing the geometric distortions between a parent  $b \in B$  and a pair of children  $(b_1, b_2) \in PCH$  by the following formulas, where we denote  $c, c_1, c_2$ , the centers of cells  $b, b_1, b_2$  and  $A, A_1, A_2$  their long axes, respectively: (i) center distortion  $\text{cen}(b, b_1, b_2) = \|c - (c_1 + c_2)/2\|$ , (ii) size distortion  $\text{siz}(b, b_1, b_2) = ||\|A\| - (\|A_1\| + \|A_2\|)|$ , and (iii) angle distortion

$$\text{ang}(b, b_1, b_2) = \text{angle}(A, A_1) + \text{angle}(A, A_2) + \text{angle}(A, c_2 - c_1).$$

Here, *angle* denotes “angles between non-oriented straight lines,” with a range from 0 to  $\pi/2$ . Introduce three positive weights  $\lambda_{\text{cen}}, \lambda_{\text{siz}}, \lambda_{\text{ang}}$  (to be estimated), and for every short lineage  $(b, b_1, b_2) \in SHLIN$  define its *distortion cost* by

$$\text{dist}(b, b_1, b_2) = \lambda_{\text{cen}} \text{cen}(b, b_1, b_2) + \lambda_{\text{siz}} \text{siz}(b, b_1, b_2) + \lambda_{\text{ang}} \text{ang}(b, b_1, b_2). \quad (7)$$

For each plausible pair of children  $(b_1, b_2) \in PCH$ , we will compute the most likely *parent cell*  $b^* = \text{parent}(b_1, b_2)$  as the cell  $b^* \in B$  minimizing  $\text{dist}(b, b_1, b_2)$  in (7) over all  $b \in B$ , as summarized by the formula

$$b^* = \text{parent}(b_1, b_2) = \underset{\{b \in B | (b, b_1, b_2) \in SHLIN\}}{\text{argmin}} \quad \text{dist}(b, b_1, b_2). \quad (8)$$

To force this minimization to yield a reliable estimate of  $b^* = \text{parent}(b_1, b_2)$  for most true pairs of children  $(b_1, b_2)$ , we calibrate the weights  $\lambda_j, j \in \{\text{cen}, \text{siz}, \text{ang}\}$  by the algorithm outlined in Section 3.2, using as “ground truth set” a fairly small set of visually identified true short lineages (parent, children). For fixed  $(b_1, b_2)$ , the set of potential parent cells  $b \in B$  has very *small size* due to the constraint (6). Hence, brute force minimization of the functional  $\text{dist}(b, b_1, b_2)$  in (7) over all  $b \in B$  such that  $(b, b_1, b_2) \in SHLIN$ , is a *fast computation* for each  $(b_1, b_2)$  in *PCH*. The distortion minimizing  $b = b^*$  yields the most likely parent cell  $\text{parent}(b_1, b_2) = b^*$ . The brute force minimization in  $b$  of  $\text{dist}(b, b_1, b_2)$  is still a greedy minimization in the sense that other soft constraints introduced further on are not taken into consideration during this preliminary fast computation of  $b^*$ .

### 3.3.3. Penalties to Enforce Adequate Parent–Children Links

Any true pair of children cells  $pch = (b_1, b_2)$  should belong to *PCH*, but must also verify lineage and geometric constraints which we now enforce via several penalties. Note that the new penalties introduced here are fully distinct from the three penalties specified above to define  $\text{dist}(b, b_1, b_2)$ .

**“Lineage” Penalty.** Valid children pairs  $(b_1, b_2) \in PCH$  should be correctly matchable with their most likely parent cell  $b^* = \text{parent}(b_1, b_2)$  (see (8)). Thus, we define the “lineage” penalty  $\text{lin}(b_1, b_2) = \text{dist}(b^*, b_1, b_2)$  by

$$\text{lin}(b_1, b_2) = \underset{\{b \in B | (b, b_1, b_2) \in shlin\}}{\text{argmin}} \quad \text{dist}(b, b_1, b_2) = \text{dist}(\text{parent}(b_1, b_2), b_1, b_2).$$

Notice that the computation of  $\text{lin}(b_1, b_2)$  is quite fast.

**“Gap” Penalty.** Denote  $\text{tips}(b)$  as the set of two endpoints of any cell  $b$ . For any pair  $pch = (b_1, b_2) \in PCH$ , define endpoints  $x_1 \in \text{tips}(b_1), x_2 \in \text{tips}(b_2)$  and the *gap* penalty  $\text{gap}(b_1, b_2)$  by

$$\text{gap}(b_1, b_2) = \|x_1 - x_2\| = \min\{\|x - y\| \text{ for } (x, y) \in TIPS\} \quad (9)$$

with  $TIPS = \text{tips}(b_1) \times \text{tips}(b_2)$ .

**“Dev” Penalty.** For rod-shaped bacteria, a true pair  $(b_1, b_2) \in PCH$  of just born children must have a small  $\text{gap}(b_1, b_2) = \|x_1 - x_2\|$  and roughly aligned cells  $b_1$  and  $b_2$ . For  $(b_1, b_2) \in PCH$ , we quantify the *deviation from alignment*  $\text{dev}(b_1, b_2)$  as follows. Let  $x_1, x_2$  be the closest endpoints of  $b_1, b_2$  (see (9)). Let  $\text{str}_{12}$  be the straight line linking the centers  $c_1, c_2$  of  $b_1, b_2$ . Let  $d_1, d_2$  be the distances from  $x_1, x_2$  to the line  $\text{str}_{12}$ . Then, set

$$\text{dev}(b_1, b_2) = \frac{d_1 + d_2}{\|c_2 - c_1\|}.$$

**“Ratio” Penalty.** True children pairs must have nearly equal lengths. Thus, for  $(b_1, b_2) \in PCH$  with lengths  $L_1, L_2$ , we define the length *ratio penalty* by

$$\text{ratio}(b_1, b_2) = |(L_1/L_2) + (L_2/L_1) - 2|.$$

**“Rank” Penalty.** Let  $L_{\min}$  be the minimum cell length over all cells in  $B_+$ . In  $B_+$ , children pairs  $(b_1, b_2)$  just born during interframe  $J \rightarrow J_+$  must have lengths  $L_1, L_2$  close to  $L_{\min}$ . Thus, for  $(b_1, b_2) \in PCH$ , we define the *rank* penalty by

$$\text{rank}(b_1, b_2) = |(L_1/L_{\min}) - 1| + |(L_2/L_{\min}) - 1|.$$

Given two successive images  $J, J_+$ , we seek the set  $X = \text{trueCH}$  of true children pairs in  $B_+ \times B_+$ , which is an unknown subset of  $PCH$ . In Section 3.5 below, we replace  $X$  by its indicator function  $z$  and we build a cost function  $E(z)$  which should be nearly minimized when  $z$  is close to the indicator of  $\text{trueCH}$ . A key term of  $E(z)$  will be a weighted linear combination of the penalty functions  $\{\text{lin}, \text{gap}, \text{dev}, \text{ratio}, \text{rank}\}$ . Since these penalties are different from those introduced in Section 3.3.2, we estimate their weights in the cost function  $E(z)$  by the algorithm outlined in Section 3.2. The minimization of  $E(z)$  will be implemented by simulations of a BM with energy function  $E(z)$ . We present these stochastic neural networks in the next section.

### 3.4. Generic Boltzmann Machines (BMs)

Minimization of our main cost functionals is a heavily combinatorial task, since the unknown variable is a mapping between two finite sets of sizes ranging from 80 to 120. To handle these minimizations, we use BMs originally introduced by Hinton et al. (see [93,94]). Indeed, these recurrent stochastic neural networks can efficiently emulate some forms of simulated annealing.

Each BM implemented here is a network  $BM = \{U_1, \dots, U_N\}$  of  $N$  *stochastic neurons*  $U_j$ . In the BM context, the time  $t = 0, 1, 2, \dots$  is discretized and represents the number of steps in a Markov chain, where the successive updates  $Z(t) \rightarrow Z(t+1)$  of the BM configuration  $Z(t)$  are analogous to the steps of a Gibbs sampler. The *configuration*  $Z(t) = \{Z_1(t), \dots, Z_N(t)\}$  of the whole network  $BM$  at time  $t$  is defined by the random states  $Z_j(t)$  of all neuron  $U_j$ . Each  $Z_j(t)$  belongs to a fixed finite set  $W(j)$ . Hence,  $Z(t)$  belongs to the *configurations* set  $\text{CONF} = W(1) \times \dots \times W(N)$ .

Neuron interactivity is specified by a finite set  $\text{CLQ}$  of *cliques*. Each clique  $K$  is a subset of  $S = \{1, \dots, N\}$ . During configuration updates  $Z(t) \rightarrow Z(t+1)$ , neurons may interact only if they are in the same clique. Here, all cliques  $K$  are of small sizes 1, or 2, or 3.

For each clique  $K$ , one specifies an energy function  $J_K(z)$  defined for all  $z \in \text{CONF}$ , with  $J_K(z)$  depending only on the  $z_j$  such that  $j \in K$ . The full energy  $E(z)$  of configuration  $z$  is then defined by

$$E(z) = \sum_{K \in \text{CLQ}} J_K(z).$$

The BM stochastic dynamics  $Z(t) \rightarrow Z(t+1)$  is driven by the energy function  $E(z)$ , and by a fixed decreasing sequence of *virtual temperatures*  $\text{Temp}(t) > 0$ , tending slowly to 0 as  $t \rightarrow \infty$ . Here, we use standard temperature schemes of the form  $\text{Temp}(t) \equiv c\eta^t$  with fixed  $c > 0$  and slow *decay rate*  $0.99 < \eta < 1$ .

We have implemented the classical “*asynchronous*” BM dynamics. At each time  $t$ , *only one* random neuron  $U_j$  may modify its state, after reading the states of all neurons belonging to cliques containing  $U_j$ . A much faster alternative, implementable on GPUs, is the “*synchronous*” BM dynamics, where at each time  $t$  roughly 50% of all neurons may simultaneously modify their states (see [95–97]). The detailed BM dynamics are presented in the appendix (see Appendix A).

When the virtual temperatures  $\text{Temp}(t)$  decrease slowly enough to 0, the energy  $E(Z(t))$  converges in probability to a local minimum of the BM energy  $E(z)$  over all configurations  $z \in \text{CONF}$ .

### 3.5. Optimized Set of Parent–Children Triplets

Next, we formulate the search for bona fide parent–children triplets as an optimization problem. For brevity, this outline is restricted to situations where (3) holds, as is the case for our synthetic image data. Simple modifications extend this approach to the relaxed constraint (4), which we used for lab videos of live cell populations. Fix successive images  $J, J_+$  with a positive number of cell divisions  $\text{DIV} = N_+ - N$ . Denote  $PCH = \{pch_1, pch_2, \dots, pch_m\}$  the set of  $m$  plausible children pairs  $(b_1, b_2)$  in  $B_+$ . The penalties *lin*, *gap*, *dev*, *ratio*, and *rank* defined above for all pairs  $(b_1, b_2) \in PCH$  determine five numerical vectors  $LIN, GAP, DEV, RAT, RANK$  in  $\mathbb{R}^m$  with coordinates  $LIN_j = \text{lin}(pch_j)$ ,  $GAP_j = \text{gap}(pch_j)$ ,  $DEV_j = \text{dev}(pch_j)$ ,  $RAT_j = \text{ratio}(pch_j)$ ,  $RANK_j = \text{rank}(pch_j)$ .

We now define a *binary* BM constituted by  $m$  *binary* stochastic neurons  $U_j$ ,  $j = 1 \dots m$ . At time  $t = 0, 1, 2, \dots$ , each  $U_j$  has a random *binary valued state*  $Z_j(t) = 1$  or 0. The random configuration  $Z(t) = [Z_1(t), \dots, Z_m(t)]$  of this BM belongs to the configuration space  $\text{CONF} = \{0, 1\}^m$  of all binary vectors  $z = [z_1, \dots, z_m]$ . Let  $SUB$  be the set of all subsets of  $PCH$ . Each configuration  $z \in \text{CONF}$  is the indicator function of a subset  $\text{sub}(z)$  of  $PCH$ . We view each  $\text{sub}(z) \in SUB$  as a possible estimate for the unknown set  $\text{trueCH} \subset B_+ \times B_+$  of true children pairs  $(b_1, b_2)$ . For each potential estimate  $\text{sub}(z)$  of  $\text{trueCH}$ , the “*lack of quality*” of the estimate  $\text{sub}(z)$  will be penalized by the *energy function*  $E(z) \geq 0$  of our binary BM. We now specify the energy  $E(z)$  for all  $z \in \text{CONF}$  by combining the penalty terms introduced above. Note that the penalty terms introduced in Section 3.3.2 are quite different from those introduced in Section 3.3.3. No cell in  $B_+$  can be assigned to more than one parent in  $b$ . To enforce this constraint, define the symmetric  $m \times m$  binary matrix  $[Q_{j,k}]$  by (i)  $Q_{j,k} = 1$  if  $j \neq k$  and the two pairs  $pch_j, pch_k$  have one cell in common, (ii)  $Q_{j,k} = 0$  if  $j \neq k$  and the two pairs  $pch_j, pch_k$  have no cell in common, (iii)  $Q_{j,j} = 0$  for all  $j$ .

The quadratic penalty  $z \mapsto \langle z, Qz \rangle$  is non-negative for  $z \in \text{CONF}$ , and must be zero if  $\text{sub}(z) = \text{trueCH}$ . Introduce six positive weight parameters to be selected further on  $\lambda_j$ ,  $j \in \{\text{lin}, \text{gap}, \text{dev}, \text{rat}, \text{rank}, Q\}$ . Define the vector  $V \in \mathbb{R}^m$  as a weighted linear combination of the penalty vectors  $LIN, GAP, DEV, RAT, RANK$

$$V = \lambda_{\text{lin}} LIN + \lambda_{\text{gap}} GAP + \lambda_{\text{dev}} DEV + \lambda_{\text{rat}} RAT + \lambda_{\text{rank}} RANK.$$

For any configuration  $z \in \text{CONF}$ , the BM energy  $E(z)$  is defined by the *quadratic function*

$$E(z) = \langle V, z \rangle + \lambda_Q \langle z, Qz \rangle. \quad (10)$$

We already know that the unknown set  $trueCH$  of true children pairs must have cardinal  $DIV = N_+ - N$ . Thus, we seek a configuration  $z^* \in CONF$  minimizing the energy  $E(z)$  under the rigid constraint  $\text{card}\{sub(z)\} = DIV$ . Let  $ONE \in \mathbb{R}^m$  be the vector with all its coordinates equal to 1. The constraint on  $z$  can be reformulated as  $\langle ONE, z \rangle = DIV$ . We want the unknown  $trueCH$  to be close to the solution  $z^*$  of the constrained minimization problem

$$z^* = \underset{z \in CONF}{\operatorname{argmin}} E(z) \quad \text{subject to} \quad \langle ONE, z \rangle = DIV.$$

To force this minimization to yield a reliable estimate of  $trueCH$ , we calibrate the six weights

$$\lambda_j, j \in \{\text{lin}, \text{gap}, \text{dev}, \text{rat}, \text{rank}, Q\}$$

by the algorithm in Section 3.2. Denote  $CONF_1$  the set of all  $z \in CONF$  such that  $\langle ONE, z \rangle = DIV$ . To minimize  $E(z)$  under the constraint  $z \in CONF_1$ , fix a slowly decreasing temperature scheme  $Temp(t)$  as in Section 3.4. We need to force the BM stochastic configurations  $Z(t)$  to remain in  $CONF_1$ . Then, for large time step  $t$ , the  $Z(t)$  will converge in probability to a configuration  $z^* \in CONF_1$  approximately minimizing  $E(z)$  under the constraint  $z \in CONF_1$ .

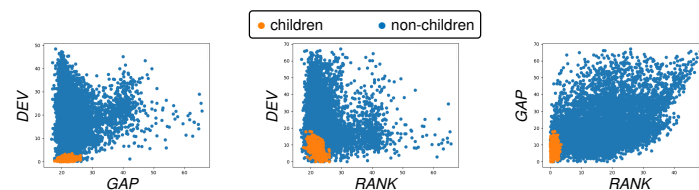
Start with any  $Z(0) \in CONF_1$ . Assume that, for  $0 \leq s \leq t$ , one has already dynamically generated BM configurations  $Z(s) \in CONF_1$ . Then, randomly select two sites  $j, k$  such that  $Z_j(t) = 1$  and  $Z_k(t) = 0$ . Compute a virtual configuration  $Y$  by setting  $Y_j = 0$ ,  $Y_k = 1$ , and  $Y_i \equiv Z_i$  for all sites  $i$  different from  $j$  and  $k$ . Compute the energy change  $\Delta E = E(Y) - E(Z(t))$ , and the probability  $p(t) = \exp(-D/Temp(t))$ , where  $D = \max\{0, \Delta E\}$ . Then, randomly select  $Z(t+1) = Y$  or  $Z(t+1) = Z(t)$  with respective probabilities  $p(t)$  and  $(1 - p(t))$ . Clearly, this forces  $Z(t+1) \in CONF_1$ .

### 3.6. Performance of Automatic Children Pairing on Synthetic Videos

In the following subsections, we provide experimental results for pairing children and parent cells.

#### 3.6.1. Children Pairing: Fast BM Simulations

For  $m = \text{card}(PCH) \leq 1000$ , one can reduce the computational cost for BM dynamics simulations by pre-computing and storing the  $m \times m$  symmetric binary matrix  $Q$ , as well as the  $m$ -dimensional vectors  $LIN, GAP, DEV, RAT, RANK$  and their linear combination  $V$ . A priori reduction of  $m$  significantly reduces the computing times, and can be implemented by trimming away the pairs  $pch_j \in PCH$  for which the penalties  $LIN_j, GAP_j, DEV_j, RAT_j$ , and  $RANK_j$  are all larger than predetermined empirical thresholds. We performed a study on 100 successive (synthetic) images. We show scatter plots for the most informative penalty terms in Figure 3. These plots allow us to determine adequate thresholds for the penalty terms. We observed that, for the synthetic and real data, we considered the trimming of  $DEV, GAP$ , and  $RANK$  reduced the percentage of invalid children pairs by 95%, therefore drastically reducing the combinatorial complexity of the problem.



**Figure 3.** Scatter plots for tandems of the penalty terms  $DEV, GAP$ , and  $RANK$ . We mark in orange the true children pairs and in blue invalid children pairs. These plots allow us to identify appropriate empirical thresholds to trim the (considered synthetic) data in order to reduce the computational complexity of the parent–children pairing.



The quadratic energy function  $E(z)$  is the sum of clique energies  $J_K(z)$  involving only cliques of cardinality 1 and 2. For any clique  $K = \{j\}$  of cardinality 1, with  $1 \leq j \leq m$ , one has  $J_K(z) = V_j z_j$ . For any clique  $K = \{j, k\}$  of cardinality 2, with  $1 \leq j < k \leq m$ , one has  $J_K(z) = 2Q_{j,k} z_j z_k$ . A key computational step when generating  $Z(t+1)$  is to evaluate the energy change  $\Delta E$  when one flips the binary values  $Z_j(t) = 1$  and  $Z_k(t) = 0$  by the new value  $(1 - z_i)$  for a fixed single site  $i$ . This step is quite fast since it uses only the numbers  $V_j$ ,  $V_k$ , and  $\langle q(j), Z(t) \rangle$ ,  $\langle q(k), Z(t) \rangle$ , where  $q(i)$  is the  $i^{\text{th}}$  row of the matrix  $Q$ .

### 3.6.2. Children Pairing: Implementation on Synthetic Videos

We have implemented our children pairing algorithms on synthetic image sequences having 100 to 500 image frames with 1 min interframe (benchmark set BENCH1; see Section 2.1). The cell motion bound  $w/2$  per interframe was defined by  $w = 20$  pixels. The parameter  $\tau$  that defines the sets  $PCH$  of plausible children pairs (see (5)) was set at  $\tau = 45$  pixels.

The known true cell registrations indicated that, in our typical BENCH1 image sequence, the successive sets  $PCH$  had average cardinalities of 120, while the number of true children pairs per  $PCH$  roughly ranged from 2 to 6 with a median of 4. The size of the reduced configuration space  $CONF1$  per image frame thus ranged from  $10^4$  to  $120^6/6! = 4.2 \times 10^9$  with a median of  $9 \times 10^6$ .

Our weights estimation technique introduced in Section 3.2 yields the weights

$$[\lambda_{\text{cen}}, \lambda_{\text{siz}}, \lambda_{\text{ang}}] = [0.255, 0.05, 0.05]$$

and

$$[\lambda_{\text{gap}}, \lambda_{\text{dev}}, \lambda_{\text{rat}}, \lambda_{\text{rank}}] = [0.01, 1, 0.0001, 0.05]$$

for the penalties introduced in Section 4. To reduce the computing time for hundreds of BM energy minimizations on the BENCH1 image sequences, we excluded obviously invalid children pairs in each  $PCH$  set, by simultaneously thresholding of the penalty terms. The BM temperature scheme was  $Temp(t) = 1000 (0.995)^t$ , with the number of epochs capped at 5000. The average CPU time for BM energy minimization dedicated to optimized children pairing was about 30 seconds per frame. (We provide hardware specifications in Appendix B).

### 3.6.3. Parent–Children Matching: Accuracy on Synthetic Videos

For each successive image pair  $J, J_+$ , with cells  $B, B_+$  of cardinality  $N < N_+$ , our parent–children matching algorithm computes a set  $SHL$  of short lineages  $(b, b_1, b_2)$ , where the cell  $b \in B$  is expected to be the parent of cells  $b_1, b_2 \in B_+$ . Recall that  $DIV = N_+ - N$  provides the number of cell divisions during the interframe  $J \rightarrow J_+$ . The number  $VAL$  of correctly reconstructed short lineages  $(b, b_1, b_2) \in SHL$  is obtained by direct comparison to the known ground truth registration  $J \rightarrow J_+$ . For each frame  $J$ , we define the *pcp-accuracy* of our parent–children pairing algorithm as the ratio  $VAL/DIV$ .

We have tested our parent–children matching algorithm on three long synthetic image sequences BENCH1 (500 frames), BENCH2 (300 frames), and BENCH3 (300 frames), with respective interframes of 1, 2, and 3 min. For each frame  $J_k$ , we computed the *pcp-accuracy* between  $J_k$  and  $J_{k+1}$ .

We report the accuracies of our parent–children pairing algorithms in Table 2. For BENCH1, all 500 *pcp-accuracies* reached 100%. For BENCH2, *pcp-accuracies* reach 100% for 298 frames out of 300, and for the remaining two frames, accuracies were still high at 93% and 96%. For BENCH3, where interframe duration was longest (3 min), the 300 *pcp-accuracies* decreased slightly but still averaged 99%, and never fell below 90%.

**Table 2.** Accuracies of parent–children pairing algorithm. We applied our parent–children pairing algorithm to three long synthetic image sequences BENCH1 (500 frames), BENCH2 (300 frames), and BENCH3 (300 frames), with interframe intervals of 1, 2, 3 min, respectively. The table summarizes the resulting pcg-accuracies. Note that pcg-accuracies are practically always at 100%. For BENCH2, pcg-accuracies are 100% for 298 frames out of 300, and for the remaining two frames, accuracies were still high at 93% and 96%. For BENCH3, the average pcg-accuracy for the 3 min interframe is 99%.

Sequence	Pcg-Accuracy	Frames
BENCH1	$acc = 100\%$	500 out of 500
BENCH2	$acc = 100\%$	298 out of 300
BENCH2	$99\% \geq acc \geq 93\%$	2 out of 300
BENCH3	$acc = 100\%$	271 out of 300
BENCH3	$99\% \geq acc \geq 95\%$	17 out of 300
BENCH3	$94\% \geq acc \geq 90\%$	12 out of 300

### 3.7. Reduction to Registrations with No Cell Division

Fix successive frames  $J, J_+$  and their cell sets  $B, B_+$ . We seek the unknown registration mapping  $f : B \rightarrow \{B_+ \cup (B_+ \times B_+)\}$ , where  $f(b) \in B_+$  iff cell  $b$  did not divide during the interframe  $J \rightarrow J_+$  and  $f(b) = (b_1, b_2) \in B_+ \times B_+$  iff cell  $b$  divided into  $(b_1, b_2)$  during the interframe.

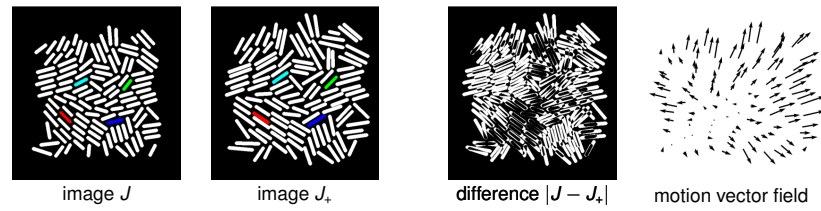
If  $\text{card}(B) = N < N_+ = \text{card}(B_+)$ , we know that the number of cell divisions during the interframe  $J \rightarrow J_+$  should be  $DIV = DIV(B, B_+) = N_+ - N > 0$ . We then apply the parent–children matching algorithm outlined above to compute a set  $SHL = SHL(B, B_+)$  of short lineages  $(b, b_1, b_2)$  with  $b \in B, b_1, b_2 \in B_+$  and  $\text{card}(SHL) = DIV$ . For each  $(b, b_1, b_2) \in SHL$ , the cell  $b$  is computed by  $b = \text{parent}(b_1, b_2)$  as the parent cell of the two children cells  $b_1, b_2 \in B_+$ .

For each  $(b, b_1, b_2) \in SHL$ , eliminate from  $B$  the parent cell,  $b$ , and eliminate from  $B_+$  the two children cells  $b_1, b_2$ . We are left with two residual sets,  $resB \subset B$  and  $resB_+ \subset B_+$ , having the same cardinality,  $N - DIV = N_+ - 2DIV$ . Assuming that our set  $SHC$  of short lineages is correctly determined, the cells  $b \in resB$  should not divide in the interframe  $J \rightarrow J_+$ , and hence have a single (still unknown) registration  $f(b) \in resB_+$ . Thus, the still unknown part of the registration  $f$  is a bijection from  $resB$  to  $resB_+$ .

Let  $divB = B - resB$  and  $divB_+ = B_+ - resB_+$ . For each  $b \in divB$ , the cell  $b$  divides into the unique pair of cells,  $(b_1, b_2) \in divB_+ \times divB_+$ , such that  $(b, b_1, b_2) \in SHL$ . Hence, we can set  $f(b) = (b_1, b_2)$  for all  $b \in divB$ . Thus, the remaining problem to solve is to compute the bijective registration  $f : resB \rightarrow resB_+$ . We have reduced the registration discovery to a new problem, where *no cell divisions occur* in the interframe duration. In what follows, we present our algorithm to solve this registration problem.

### 3.8. Automatic Cell Registration after Reduction to Cases with No Cell Division

As indicated above, we can *explicitly reduce* the generic cell tracking problem to a problem where there is *no cell division*. We consider images  $J, J_+$  with associated cell sets  $B, B_+$  such that  $N = \text{card}(B) = \text{card}(B_+)$ . Hence, there are no cell divisions in the interframe  $J \rightarrow J_+$  and the map  $f$  of this reduced problem is (in principle) a bijection  $f : B \rightarrow B_+$  with  $\text{card}(B) = \text{card}(B_+)$ . In Figure 4, we show two typical successive images we use for testing with no cell division generated by the simulation software [12,81] (see Section 2.1).



**Figure 4.** Simulated cell dynamics. From left to right, two successive simulated images  $J$  and  $J_+$  with an interframe time of six minutes and no cell division, their image difference  $|J - J_+|$ , and the associated motion vectors. For the image  $J$  and  $J_+$  we color four pairs of cells in  $B \times B_+$ , which should be matched by the true cell registration mapping. Notice that the motion for an interframe time of six minutes is significant. We can observe that, even without considering cell division, we can no longer assume that corresponding cells in frame  $J$  and  $J_+$  overlap.

### 3.8.1. The Set $MAP$ of Many-to-One Cell Registrations

We have reduced the registration search to a situation where, during the interframe  $J \rightarrow J_+$ , no cell has divided, no cell has disappeared, and no cell has suddenly emerged in  $B_+$  without originating from  $B$ . The unknown registration  $f : B \rightarrow B_+$  should then in principle be injective and onto. However, for computational efficiency, we will temporarily relax the bijectivity constraint on  $f$ . We will seek  $f$  in the set  $MAP$  of all *many-to-one mappings*  $f : B \rightarrow B_+$  such that for each  $b \in B$ , the cell  $f(b)$  is in the target window  $W(b) \subset B_+$  (see Section 2.3).

### 3.8.2. Registration Cost Functional

To design a cost functional  $\text{cost}(f)$ , which should be roughly minimized when  $f \in MAP$  is very close to the true registration from  $B$  to  $B_+$ , we linearly combine penalties  $\text{match}(f)$ ,  $\text{over}(f)$ ,  $\text{stab}(f)$ ,  $\text{flip}(f)$  weighted by unknown positive weights  $\lambda_{\text{match}}$ ,  $\lambda_{\text{over}}$ ,  $\lambda_{\text{stab}}$ ,  $\lambda_{\text{flip}}$ , to write, for all registrations  $f \in MAP$ ,

$$\text{cost}(f) = \lambda_{\text{match}} \text{match}(f) + \lambda_{\text{over}} \text{over}(f) + \lambda_{\text{stab}} \text{stab}(f) + \lambda_{\text{flip}} \text{flip}(f). \quad (11)$$

We specify the individual terms that appear in (11) below. Ideally, the minimizer of  $\text{cost}(f)$  over all  $f \in MAP$  is close to the unknown true registration mapping  $f : B \rightarrow B_+$ . To enforce a good approximation of this situation, we first estimate efficient positive weights by applying our calibration algorithm (see Section 3.2). The actual minimization of  $\text{cost}(f)$  over all  $f \in MAP$  is then implemented by a BM described in Section 3.9.

**Cell Matching Likelihood:**  $\text{match}(f)$ . Here, we extend a pseudo likelihood approach used to estimate parameters in Markov random fields modeling by Gibbs distributions (see [98]). Recall that  $g.\text{rate}$  is the *known* average cell growth rate. For any cells  $b \in B$ ,  $b_+ \in B_+$ , the geometric quality of the matching  $b \mapsto b_+$  relies on three main characteristics: (i) motion  $c(b_+) - c(b)$  of the cell center  $c(b)$ , (ii) angle between the long axes  $A(b)$  and  $A(b_+)$ , (iii) cell length ratio  $\|A(b_+)\| / \|A(b)\|$ . Thus, for all  $b \in B$  and  $b_+$  in the target window  $W(b)$ , define (i) Kinetic energy:  $\text{kin}(b, b_+) = \|c(b) - c(b_+)\|^2$ . (ii) Distortion of cell length:  $\text{dis}(b, b_+) = |\log(\|A(b_+)\| / \|A(b)\|) - \log g.\text{rate}|^2$ . (iii) Rotation angle:  $0 \leq \text{rot}(b, b_+) \leq \pi/2$  is the geometric angle between the straight lines carrying  $A(b)$  and  $A(b_+)$ .

Fix  $b \in B$ , and let  $b'$  run through the whole target window  $W(b)$ . The finite set of values thus reached by the kinetic penalties  $\text{kin}(b, b')$  has two smallest values  $\text{kin}_1(b)$ ,  $\text{kin}_2(b)$ . Define  $\text{list.kin} = \bigcup_{b \in B} \{\text{kin}_1(b), \text{kin}_2(b)\}$ , which is a list of  $2N$  “low” kinetic penalty values. Repeat this procedure for the penalties  $\text{dis}(b, b')$  and  $\text{rot}(b, b')$  to similarly define a  $\text{list.dis}$  of  $2N$  “low” distortion penalty values, and a  $\text{list.rot}$  of  $2N$  “low” rotation penalty values.

The three sets  $\text{list.kin}$ ,  $\text{list.dis}$ ,  $\text{list.rot}$  can be viewed as three random samples of size  $2N$ , respectively, generated by three unknown probability distributions  $P_{\text{kin}}$ ,  $P_{\text{dis}}$ ,  $P_{\text{rot}}$ . We approximate these three probabilities by their *empirical* cumulative distribution functions

$CDF_{kin}$ ,  $CDF_{dis}$ ,  $CDF_{rot}$ , which can be readily computed. We now use the right tails of these three CDFs to compute separate probabilistic evaluations of how *likely* the matching of cell  $b \in B$  with cell  $b_+ \in W(b)$  is. For any fixed mapping  $f \in MAP$ , and any  $b \in B$ , set  $b_+ = f(b)$ . Compute the three penalties  $vkin = kin(b, b_+)$ ,  $vdis = dis(b, b_+)$ ,  $vrot = rot(b, b_+)$ , and define three associated “likelihoods” for the matching  $b \rightarrow b_+ = f(b)$ :

$$\begin{aligned} LIK_{kin}(b, b_+) &= 1 - CDF_{kin}(vkin), \\ LIK_{dis}(b, b_+) &= 1 - CDF_{dis}(vdis), \\ LIK_{rot}(b, b_+) &= 1 - CDF_{rot}(vrot). \end{aligned}$$

High values of the penalties  $vkin$ ,  $vdis$ ,  $vrot$  thus will yield three small likelihoods for the matching  $b \rightarrow b_+ = f(b)$ . With this, we can define a “joint likelihood”  $0 \leq LIK(b, b_+) \leq 1$  evaluating how likely is the matching  $b \rightarrow b_+ = f(b)$ :

$$LIK(b, b_+) = \prod_{j \in \{kin, dis, rot\}} LIK_j(b, b_+). \quad (12)$$

Note that higher values of  $LIK(b, b_+)$  correspond to a better geometric quality for the matching of  $b$  with  $b_+ = f(b)$ . To avoid vanishingly small likelihoods, whenever  $LIK(b, b_+) < 10^{-6}$ , we replace it by  $10^{-6}$ . Then, for any mapping  $f \in MAP$ , we define its *likelihood*  $lik(f)$  by the finite product

$$lik(f) = \prod_{b \in B} LIK(b, f(b)).$$

The product of these  $N$  likelihoods is typically very small, since  $N = \text{card}(B)$  can be large. Thus, we evaluate the geometric matching quality  $\text{match}(f)$  of the mapping  $f$  via the averaged *log-likelihood* of  $f$ , namely,

$$\text{match}(f) = -\frac{1}{N} \log lik(f) = -\frac{1}{N} \sum_{b \in B} \log LIK(b, f(b)).$$

Good registrations  $f \in MAP$  should yield small values for the criterion  $\text{match}(f)$ .

**Overlap:**  $\text{over}(f)$ . We expect *bona fide* cell registrations  $f \in MAP$  to be bijections. Consequently, we want to penalize mappings  $f$  which are many-to-one. We say that two distinct cells  $(b, b') \in B \times B$  do *overlap* for the mapping  $f \in MAP$  if  $f(b) = f(b')$ . The total number of overlapping pairs  $(b, b')$  for  $f$  defines the *overlap penalty*:

$$\text{over}(f) = \frac{1}{\text{card}(B)} \sum_{b \in B} \sum_{b' \in B} 1_{f(b)=f(b')}.$$

**Neighbor Stability:**  $\text{stab}(f)$ . Let  $B = \{b_1, \dots, b_N\}$ . Denote  $G_i$  as the set of all neighbors for cell  $b_i$  in  $B$  (i.e.,  $b_j \sim b_i \iff b_j \in G_i$ ; see Section 2.3). For *bona fide* registrations  $f \in MAP$ , and for most pairs of neighbors  $b_i \sim b_j$  in  $B$ , we expect  $f(b_i)$  and  $f(b_j)$  to remain neighbors in  $B_+$ . Consequently, we penalize the lack of “neighbors stability” for  $f$  by

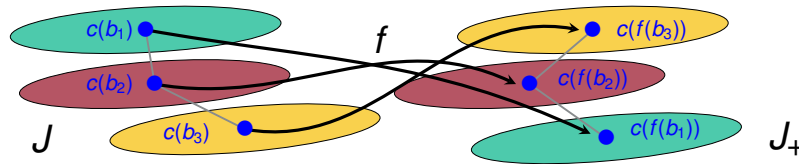
$$\text{stab}(f) = \sum_i \sum_{j \neq i} \frac{1}{N|G_i||G_j|} 1_{b_i \sim b_j} 1_{f(b_i) \not\sim f(b_j)}.$$

**Neighbor Flip:**  $\text{flip}(f)$ . Fix any mapping  $f \in MAP$ , any cell  $b \in B$  and any two neighbors  $b', b''$  of  $b$  in  $B$ . Let  $z = f(b)$ ,  $z' = f(b')$ ,  $z'' = f(b'')$ . Let  $c, c', c''$  and  $d, d', d''$  be the centers of cells  $b, b', b''$  and  $z, z', z''$ . Let  $\alpha$  be the oriented angle between  $c' - c$  and  $c'' - c$ , and let  $\alpha_f$  be the angle between  $d' - d$  and  $d'' - d$ , respectively. We say that the mapping  $f$  has *flipped* cells  $b', b''$  around  $b$ , and we set  $\text{FLIP}(f, b, b', b'') = 1$  if  $z', z''$  are both neighbors of  $z$ , and the two angles  $\alpha, \alpha_f$  have *opposite signs*. In all other cases, we set  $\text{FLIP}(f, b, b', b'') = 0$ .

For any registration  $f \in \text{MAP}$ , define the *flip penalty* for  $f$  by

$$\text{flip}(f) = \sum_{b \in B} \sum_{b' \in B} \sum_{b'' \in B} \frac{1}{N|G(b)|^2} \text{FLIP}(f, b, b', b''),$$

where  $G(b)$  is the neighborhood of cell  $b$  in  $B$ . In Figure 5, we illustrate an example of an unwanted cell flip.



**Figure 5.** Illustration of an undesirable flip for the mapping  $f$ . The cells  $b_1$  and  $b_3$  are neighbors of  $b_2$ , and mapped by  $f$  on neighbors  $z_1 = f(b_1)$ ,  $z_3 = f(b_3)$  of  $z_2 = f(b_2)$ , as should be expected for bona fide cell registrations. However, for this mapping  $f$ , we have  $z_3$  above  $z_2$  above  $z_1$ , whereas, for the original cells, we had  $b_1$  above  $b_2$  above  $b_3$ . Our cost function penalizes flips of this nature.

### 3.9. BM Minimization of Registration Cost Function

In what follows, we define the optimization problem for the registration of cells from one frame to another (i.e., cell tracking), as well as associated methodology and parameter estimates.

#### 3.9.1. BM Minimization of $\text{Cost}(f)$ over $f \in \text{MAP}$

Let  $B, B_+$  be two successive sets of cells. As outlined above, we have reduced the problem to one in which we can assume that  $N = \text{card}(B) = \text{card}(B_+)$ , so that there is no cell division during the interframe. Write  $B = \{b_1, \dots, b_N\}$ . For short, denote  $W(j) \subset B_+$  instead of  $W(b_j)$  the target window of cell  $b_j$ . We seek to minimize  $\text{cost}(f)$  over all registrations  $f \in \text{MAP}$ . Let  $\text{BM}$  be a BM with sites  $S = \{1, \dots, N\}$  and stochastic neurons  $\{U_1, \dots, U_N\}$ . At time  $t$ , the random state  $Z_j(t)$  of  $U_j$  will be some cell  $z_j$  belonging to the target window  $W(j)$  and the random configuration  $Z(t) = \{Z_1(t), \dots, Z_N(t)\}$  of the whole BM belongs to the configurations set  $\text{CONF} = W(1) \times \dots \times W(N)$ .

To any configuration  $z = \{z_1, \dots, z_N\} \in \text{CONF}$ , we associate a unique cell registration  $f \in \text{MAP}$  defined by  $f(b_j) = z_j$  for all  $j$ , denoted by  $f = \text{map}(z)$ . This determines a bijection  $z \mapsto f = \text{map}(z)$  from  $\text{CONF}$  onto  $\text{MAP}$ . The inverse of  $\text{map} : \text{CONF} \rightarrow \text{MAP}$  will be called  $\text{range} : \text{MAP} \rightarrow \text{CONF}$ , and is defined by  $z = \text{range}(f)$ , when  $z_j = f(b_j)$  for all  $j$ .

#### 3.9.2. BM Energy Function $E(z)$

We now define the energy function  $E(z) \geq 0$  of our BM for all  $z \in \text{CONF}$ . Denote  $E^* = \text{minimize}_{z \in \text{CONF}} E(z)$ . Since  $f \mapsto z = \text{range}(f)$  is a bijection from  $\text{MAP}$  to  $\text{CONF}$ , we must have

$$E^* = \text{minimize}_{z \in \text{CONF}} E(z) = \text{minimize}_{f \in \text{MAP}} E(\text{range}(f)).$$

Our goal is to minimize  $\text{cost}(f)$ , and we know that BM simulations should roughly minimize  $E(z)$  over all  $z \in \text{CONF}$ . Thus, we define the BM energy function  $E(z)$  by forcing

$$\text{cost}(f) = E(\text{range}(f)) \quad (13)$$

for any registration mapping  $f \in \text{MAP}$ , which—due to the preceding subsection—is equivalent to

$$E(z) = \text{cost}(\text{map}(z)) \quad (14)$$



for all configurations  $z \in \text{CONF}$ . The next subsection will explicitly express the energy  $E(z)$  in terms of *cliques* of neurons. Due to (13) and (14), we have

$$E^* = \underset{f \in \text{MAP}}{\text{minimize}} \text{cost}(f) = \underset{z \in \text{CONF}}{\text{minimize}} E(z).$$

For large time  $t$ , the BM stochastic configuration  $Z(t)$  tends with high probability to concentrate on configurations  $z \in \text{CONF}$ , which roughly minimize  $E(z)$ . The random registration  $F^t = \text{map}(Z(t))$  will belong to  $\text{MAP}$  and verify  $Z(t) = \text{range}(F^t)$ , so that  $E(Z(t)) = E(\text{range}(F^t)) = \text{cost}(F^t)$ . Consequently, for large  $t$ —with high probability—the random mapping  $F^t = \text{map}(Z(t))$  will have a value of the cost functional  $\text{cost}(F^t)$  close to  $\underset{f \in \text{MAP}}{\text{minimize}} \text{cost}(f)$ .

### 3.9.3. Cliques of Interactive Neurons

The BM energy function  $E(z)$  just defined turns out to involve only three sets of small cliques: (i)  $CL_1$  is the set of all singletons  $K = \{i\}$ , with  $i = 1 \dots N$ . (ii)  $CL_2$  is the set of all pairs  $K = \{i, j\}$  such that cells  $b_i$  and  $b_j$  are *neighbors* in  $B$ . (iii)  $CL_3$  is the set of all triplets  $K = \{i, j, k\}$  such that cells  $b_j$  and  $b_k$  are both *neighbors* of  $b_i$  in  $B$ . Denote  $CLQ = CL_1 \cup CL_2 \cup CL_3$  as the set of all cliques for our BM.

**Cliques in  $CL_1$ .** For each clique  $K = \{i\}$  in  $CL_1$ , and each  $z \in \text{CONF}$ , define its energy  $J_{\text{match},K}(z) = J_{\text{match},K}(z_i)$  by

$$J_{\text{match},K}(z) = -\frac{1}{N} \log \text{LIK}(b_i, z_i) \text{ for all } z \in \text{ZW},$$

where  $\text{LIK}$  is given by (12). Set  $J_{\text{match},K} \equiv 0$  for  $K$  in  $CL_2 \cup CL_3$ . For all  $z \in \text{CONF}$ , define the energy  $E_{\text{match}}(z)$  by

$$E_{\text{match}}(z) = \sum_{K \in CLQ} J_{\text{match},K}(z) = \sum_{K \in CL_1} J_{\text{match},K}(z),$$

which implies that the registration  $f = \text{map}(z)$  verifies  $\text{match}(f) = E_{\text{match}}(z)$ .

**Cliques in  $CL_2$ .** For all  $z \in \text{CONF}$ , all cliques  $K = \{i, j\}$  in  $CL_2$ , define the clique energies  $J_{\text{over},K}(z) = J_{\text{over},K}(z_i, z_j)$  and  $J_{\text{stab},K}(z) = J_{\text{stab},K}(z_i, z_j)$  by  $J_{\text{over},K}(z) = 1_{z_i=z_j}/N$  and

$$J_{\text{stab},K}(z) = \frac{1}{N|G_i||G_j|} 1_{b_j \sim b_i} 1_{z_j \not\sim z_i},$$

where  $|G_i|$  and  $|G_j|$  are the numbers of neighbors in  $B$  for cells  $z_i$  and  $z_j$ , respectively. Set  $J_{\text{over},K} = J_{\text{stab},K} \equiv 0$  for  $K$  in  $CL_1 \cup CL_3$ . Define the two energy functions

$$\begin{aligned} E_{\text{over}}(z) &= \sum_{K \in CLQ} J_{\text{over},K}(z) = \sum_{K \in CL_2} J_{\text{over},K}(z), \\ E_{\text{stab}}(z) &= \sum_{K \in CLQ} J_{\text{stab},K}(z) = \sum_{K \in CL_2} J_{\text{stab},K}(z), \end{aligned}$$

which implies that  $f = \text{map}(z)$  verifies  $\text{over}(f) = E_{\text{over}}(z)$  and  $\text{stab}(f) = E_{\text{stab}}(z)$ .

**Cliques in  $CL_3$ .** For each clique  $K = \{i, j, k\}$  in  $CL_3$ , define the clique energy  $J_{\text{flip},K}$  by

$$J_{\text{flip},K}(z) = J_{\text{flip}}^{i,j,k}(z) = \frac{1}{N|G_i|^2} \text{FLIP}(f^{i,j,k}, b_i, b_j, b_k),$$

where  $f^{i,j,k}$  is any registration mapping  $b_i, b_j, b_k$  onto  $z_i, z_j, z_k$ . The indicator  $\text{FLIP}$  was defined in Section 3.8.2. Set  $J_{\text{flip},K} \equiv 0$  for  $K$  in  $CL_1 \cup CL_2$ . Define the energy

$$E_{\text{flip}}(z) = \sum_{K \in CLQ} J_{\text{flip},K}(z) = \sum_{K \in CL_3} J_{\text{flip},K}(z),$$

which implies that  $f = F(z)$  verifies  $\text{flip}(f) = E_{\text{flip}}(z)$ .

Finally, define the clique energy  $J_K$  for all  $K \in CLQ$  by the linear combination

$$J_K = \lambda_{\text{match}} J_{\text{match},K} + \lambda_{\text{over}} J_{\text{over},K} + \lambda_{\text{stab}} J_{\text{stab},K} + \lambda_{\text{flip}} J_{\text{flip},K}.$$

Summing this relation over all  $K \in CLQ$  yields

$$\sum_{K \in CLQ} J_K = \lambda_{\text{match}} E_{\text{match}} + \lambda_{\text{over}} E_{\text{over}} + \lambda_{\text{stab}} E_{\text{stab}} + \lambda_{\text{flip}} E_{\text{flip}}. \quad (15)$$

Define then the final BM energy function  $z \mapsto E(z)$  by

$$E(z) = \sum_{K \in CLQ} J_K(z) \text{ for all } z \text{ in } CONF. \quad (16)$$

For any  $z \in CONF$ , the associated registration  $f = \text{map}(z)$  verifies  $\text{match}(f) = E_{\text{match}}(z)$ ,  $\text{over}(f) = E_{\text{over}}(z)$ ,  $\text{stab}(f) = E_{\text{stab}}(z)$ ,  $\text{flip}(f) = E_{\text{flip}}(z)$ . By weighted linear combination of these equalities, and, due to (15), we obtain for all configurations  $z \in CONF$ ,  $E(z) = \text{cost}(f)$  when  $f = \text{map}(z)$  or, equivalently, when  $z = \text{range}(f)$ .

### 3.9.4. Test Set of 100 Synthetic Image Pairs

As shown above, the minimization of  $\text{cost}(f)$  over all registrations  $f \in MAP$  is equivalent to seeking BM configurations  $z \in CONF$  with minimal energy  $E(z)$ . We have implemented this minimization of  $E(z)$  by the long-term asynchronous dynamics of the BM just defined. This algorithm was designed for the registration of image pairs exhibiting no cell division, and was, therefore, implemented after the automatic reduction of the generic registration problem, as indicated earlier. We have tested this specialized registration algorithm on a set of 100 pairs of successive images of simulated cell colonies exhibiting no cell divisions. These 100 image pairs were extracted from the benchmark set BENCH6 of synthetic image sequence described in Section 2.1. The 100 pairs of cell sets  $B, B_+$  had sizes  $N = \text{card}(B) = \text{card}(B_+)$  ranging from 80 to 100 cells. For each test pair  $B, B_+$ , each target window  $W(j)$  typically contained 30 to 40 cells. The set  $CONF$  of configurations had huge cardinality ranging from  $10^{130}$  to  $10^{160}$ . However, the average number of neighbors of a cell was around 4 to 5.

### 3.9.5. Implementation of BM Minimization for Cost( $f$ )

The numbers  $clq_1, clq_2, clq_3$  of cliques in  $CL_1, CL_2, CL_3$  have the following rough ranges  $80 \leq clq_1 \leq 100$ ,  $160 \leq clq_2 \leq 250$ , and  $450 \leq clq_3 \leq 600$ . For  $k = 1, 2, 3$ , denote  $val(k)$  the numbers of non-zero values for  $J_K(z)$  when  $z$  runs through  $CONF$  and  $K$  runs through all cliques of cardinality  $k$ . One easily checks the rough upper bounds  $val(1) < 4000$ ;  $val(2) < 200,000$ ;  $val(3) < 300,000$ . Hence, to automatically register  $B$  to  $B_+$ , one could pre-compute and store all the possible values of  $J_K(z)$  for all cliques  $K \in CL_1 \cup CL_2 \cup CL_3$  and all the configurations  $z \in CONF$ . This accelerates the key computing steps of the asynchronous BM dynamics, namely, for the evaluation of energy change  $\Delta E = E(z') - E(z)$ , when configurations  $z$  and  $z'$  differ at only one site  $j \in S$ . Indeed, the single site modification  $z_j \rightarrow z'_j$  affects only the energy values  $J_K(z)$  for the very small number  $r(j)$  of cliques  $K$ , which contain the site  $j$ . In our benchmark sets of synthetic images, one had  $r(j) < 24$  for all  $j \in S$ . Hence, the computation of  $\Delta E$  was fast since it requires retrieving at most 24 pairs of pre-computed  $J_K(z), J_K(z')$ , and evaluating the 24 differences  $J_K(z') - J_K(z)$ . Another practical acceleration step is to replace the ubiquitous computations of probabilities  $p(t) = \exp(-D/Temp(t))$  by simply testing the value  $-D/Temp(t)$  against 100 precomputed logarithmic thresholds.

In our implementation of ABM dynamics, we used virtual temperature schemes such as  $Temp(t) = 50 \cdot \rho^t$  with  $0.995 \leq \rho \leq 0.999$ . The BM simulation was stopped when the stochastic energy  $E(Z(t))$  had remained roughly stable during the last  $N$  steps. Since all

target windows  $W(j)$  had cardinality smaller 40, the initial configuration  $Z(0) = x$  was computed via

$$x_j = \operatorname{argmax}_{y \in W(j)} \operatorname{LIK}(b_j, y) \quad \text{for } j = 1, \dots, N,$$

where the likelihoods  $\operatorname{LIK}$  were defined by (12).

### 3.9.6. Weight Calibration

For the pair of successive *synthetic* images  $J, J_+$  displayed in Figure 4, we have  $N = \operatorname{card}(B) = \operatorname{card}(B_+) = 513$  cells. The ground truth registration  $f$  is known by construction; we used it to apply the weight calibration described in Section 3.2. We set the meta-parameter  $\gamma$  to  $10^{10}$  and obtained the vector of weights

$$\Lambda^* = [\lambda_{\text{match}}^*, \lambda_{\text{over}}^*, \lambda_{\text{stab}}^*, \lambda_{\text{flip}}^*] = [110, 300, 300, 290]. \quad (17)$$

These weights are *kept fixed for all the 100 pairs* of images taken from the set BENCH6. The determined weights are used in the cost function  $\operatorname{cost}(f)$  defined above. This correctly parametrized the BM energy function  $E(z)$ . We then simulated the BM stochastic dynamics to minimize the BM energy  $E(Z(t))$ .

### 3.9.7. BM Simulations

We launched 100 simulations of the asynchronous BM dynamics, one for each pair of successive images in our test set of 100 images taken from BENCH6. For each such pair, the ground truth mapping  $f : B \rightarrow B_+$  was known by construction and the stochastic minimization of the BM energy generated an estimated cells registration  $f' : B \rightarrow B_+$ . For each pair  $B, B_+$  in the considered set of 100 images, the accuracy of this automatically computed registration  $f'$  was evaluated by the percentage of cells  $b \in B$  such that  $f'(b) = f(b)$ . When  $\operatorname{card}(B) = N$ , our BM has  $N$  stochastic neurons, and the asynchronous BM dynamics proceeds by successive *epochs*. Each epoch is a sequence of  $N$  single site updates of the BM configuration. For each one of our 100 simulations of BM asynchronous dynamics, the number of epochs ranged from 250 to 450.

The average computing time was about eight minutes per epoch, which entailed a computing time ranging from 30 to 50 min for each one of our 100 automatic registrations  $f' : B \rightarrow B_+$  reported here. (We specify the hardware used to carry out these computations in Appendix B). Each image contains about 100 to 150 cells. Consequently, the runtime for the algorithm is approximately 20 s per cell for our prototype implementation. We note that this is only a rough estimate. The runtime depends on several factors, such as the number of cells in an image; the number of mother and daughter cells (i.e., how many cells divide); the size of the neighborhood of each individual cell (window size); the weights used in the cost function (which affects the number of epochs), etc. We note that the temperature scheme had not been optimized yet, so that these computing times are upper bounds. Earlier SBM studies [99–102] indicate that the same energy minimizations on GPUs could provide a computational speedup by a factor ranging between 30 and 50. We report registration accuracies in Table 3. For each pair of images in the considered set of 100 images, the accuracy of automatic registration was larger than 94.5%. The overall average registration accuracy was quite high at 99%.

**Table 3.** Registration accuracy for synthetic image sequence BENCH<sub>100</sub>. We consider 100 pairs of consecutive synthetic images taken from the benchmark dataset BENCH6. Automatic registration was implemented by BM minimization of the cost function  $\text{cost}(f)$ , which was parametrized by the vector of optimized weights  $\Lambda^*$  in (17). The average registration accuracy was 99%.

Registration Accuracy	Number of Frames
$\text{acc} = 100\%$	55 frames out of 100
$99\% \geq \text{acc} > 97\%$	40 frames out of 100
$96\% \geq \text{acc} > 94.5\%$	5 frames out of 100

## 4. Results

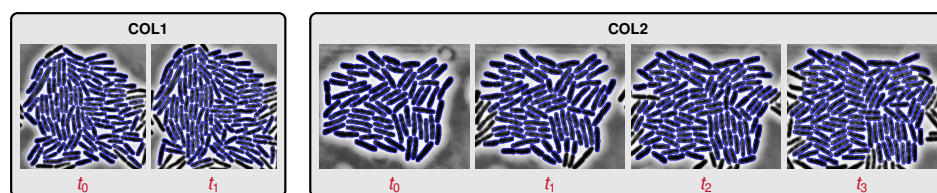
In this section, we report results for the registration for cell dynamics involving growth, motion, and cell divisions.

### 4.1. Tests of Cell Registration Algorithms on Synthetic Data

We now consider more generic long synthetic image sequences of simulated cell colonies, with a small interframe duration of one minute. We still impose the mild constraint that no cell is lost between two successive images. The main difference with the earlier benchmark of 100 images from BENCH6 is that cells are *allowed to freely divide* during interframes, as well as to grow and to move. For the full implementation on 100 pairs of successive images, we first execute the parent–children pairing, and remove the identified parent–children triplets; we can then apply our cell registration algorithmic on the reduced sets cells. Our image sequence contained 760 true parent–children triplets, which we automatically identified with an accuracy of 100%. As outlined earlier, we removed all these identified cell triplets and then applied our tracking algorithm. This left us with a total of 12,631 cells (spread over 100 frames). Full automatic registration was then implemented with an accuracy higher than 99.5%.

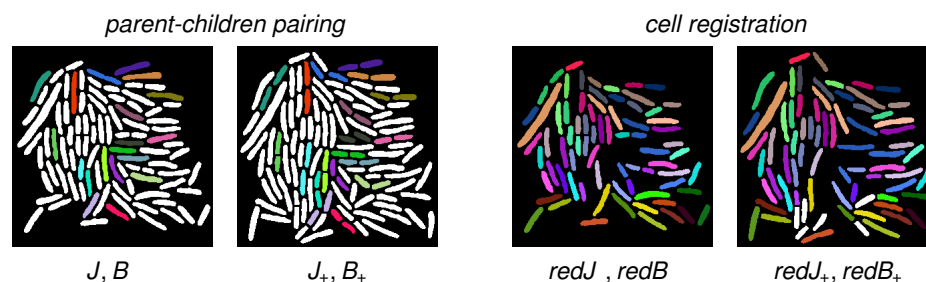
### 4.2. Tests of Cell Registration Algorithms on Laboratory Image Sequences

To test our cell tracking algorithm on pairs of consecutive images extracted from recorded image sequences of bacterial colonies (real data), we had to automatically delineate all individual cells in each image. Representative frames of these data are shown in Figure 1. We describe these data in more detail in Section 2.2. We will only briefly outline the overall segmentation approach to not distract from our main contribution—the cell tracking algorithm. We use the watershed algorithm [103] (also used, e.g., in [76]) to segment each frame into individual image segments containing one single cell each. Consequently, these regions represent over segmentations of the individual cells; we only know that each region will contain a bacteria cell  $b$ . To segment individual cells, an additional step is necessary. We then apply ad hoc nonlinear filters to remove minor segmentation artifacts. In a second step, we then identified the contour of each single cell  $b$  by applying the Mumford–Shah algorithm [104] within the image segment containing a cell  $b$ . Since this procedure is quite time-consuming for large images, we have implemented it to produce a training set of delineated individual cells to train a CNN for image segmentation. After automatic training, this CNN substantially reduces the runtime of the cell segmentation/delineation procedure. We show the resulting segmentations in Figure 6. We provide additional information regarding our approach for the segmentation of individual bacteria cells in the appendix (see Appendix D).



**Figure 6.** Segmentation results for experimental recordings of live cell colonies. We show two short image sequences extracts COL1 (left) and COL2 (right). The interframe duration is six minutes. The image sequence extract COL1 has only two successive image frames. The image sequence extract COL2 has four successive image frames. We are going to automatically compute four cell registrations, one for each pair of successive images in COL1 and COL2.

After each cell has been identified (i.e., segmented out) in each pair  $J, J_+$  of successive images, we transform  $J, J_+$  into binary images, where cells appear in white on a black background. For each resulting pair  $B, B_+$  of successive sets of cells, we apply the parent–children pairing algorithm outlined in Section 3.3 to identify all the short lineages. For the two successive images in COL1, the discovered short lineages are shown in Figure 7 (left pair of images). Here, color designates the cell triplet algorithmically identified: parent cell in image  $J$  and its two children in image  $J_+$ . We then remove each identified “parent” from  $B$  and its two children from  $B_+$ . This yields the reduced cell sets  $redB$  and  $redB_+$ . We can then apply our tracking algorithm (see Section 3.7) dedicated to situations where cells do not divide during the interframe.

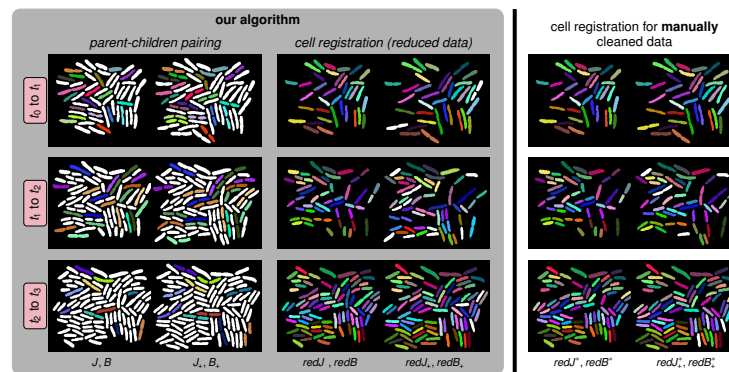


**Figure 7.** Cell tracking results for the pair COL1 of successive images  $J, J_+$  shown in Figure 6. The interframe duration is six minutes. (Left): Results for parent–children pairing on COL1. Automatically detected parent–children triplets are displayed in the same color. (Right): Computed registration. The removal of the automatically detected parent–children triplets (see left column) generates the reduced cell sets  $redB$  and  $redB_+$ . Automatic registration of  $redB$  and  $redB_+$  is again displayed via identical color for the registered cell pairs  $(b, b_+)$ . Mismatches are mostly due to previous errors in parent–children pairing (see Figure 8 for a more detailed assessment).

For image sequences of live cell colonies, we had to re-calibrate most of our weight parameters. The weight parameters used for these image sequences are summarized in Table 4.

The BM temperature scheme was  $Temp(t) = 2000(0.995)^t$ , with the number of epochs capped at 5000. We illustrate our COL1 automatic registration results in Figure 7 (right pair of images). Here, if cell  $b \in redB$  has been automatically registered onto cell  $b_+ \in redB_+$ ,  $b, b_+$  share the same color. The cells colored in white in  $redB_+$  are cells which the registration algorithm did not succeed in matching to some cell in  $redB$ . These errors can essentially be attributed to errors in the parent–children pairing step. By visual inspection, we have determined that there are 14 true parent–children triplets in the successive images of COL1. Our parent–children pairing algorithm did correctly identify 11 of these 14 triplets. To check further the performance of our registration algorithm on live images, we also report automatic registration results for “manually prepared” true versions of  $redB$  and  $redB_+$ , obtained by removing “manually” the true parent–children triplets determined by visual inspection. For the short image sequence COL2, results are displayed in Figure 8.





**Figure 8.** Cell tracking results for the short image sequence COL2 in Figure 6. The interframe duration for COL2 is six minutes. COL2 involves four successive images  $J(t_i)$ ,  $i = 0, 1, 2, 3$ . In our figure, each one of the three rows displays the automatic cell registration results between images  $J(t_i)$  and  $J(t_{i+1})$  for  $i = 0, 1, 2$ . We report the accuracies of parent–children pairing and of the registration in Table 5. **(Left column):** Results for parent–children pairing. Each parent–children triplet is identified by the same color for each parent cell and its two children. **(Middle column):** Display of the automatically computed registration after removing the parent–children triplets already identified in order to generate two reduced sets  $redB$  and  $redB_+$  of cells. Again, the same color is used for each pair of automatically registered cells. The white cells in  $redB_+$  are cells which could not be registered to some cell in  $redB$ . **(Right column):** To differentiate between errors induced during automatic identification of and errors generated by automatic registration between  $redB$  and  $redB_+$ , we manually removed all “true” parent–children triplets and then applied our registration algorithm to this “cleaned” (reduced) cell sets  $redB^*$  and  $redB_+^*$ .

**Table 4.** Cost function weights for parent–children pairing in the COL1 images displayed in Figure 6.

Weights	$\lambda_{cen}$	$\lambda_{siz}$	$\lambda_{ang}$	$\lambda_{gap}$	$\lambda_{dev}$	$\lambda_{rat}$	$\lambda_{rank}$	$\lambda_{over}$
Value	3	7	100	0.8	4	0.01	0.01	600

The display setup is the same: The left column shows the results of automatic parent–children pairing. The middle column illustrates the computed registration after automatic removal of the computer identified parent–children triplets. The third column displays the computed registration after “manually” removing the true parent–children triplets determined by visual inspection. Note that the overall matching accuracy can be improved if we reduce errors in the parent–children pairing. We report quantitative accuracies in Table 5. For parent–children pairing, accuracy ranges between 70% and 78%. For pure registration after correct parent–children pairing, accuracy ranges between 90% and 100%.

**Table 5.** Cell tracking accuracy for the short image sequence COL2 in Figure 6 with an interframe of six minutes. We report the ratio of correctly predicted cell matches over the total number of true cell matches and the associated percentages. The accuracy results quantify four distinct percentages of correct detections (i) for parent cells in image  $J$ , (ii) for children cells in image  $J_+$ , (iii) for parent–children triplets, and (iv) for registered pairs of cells  $(b, b_+) \in redB \times redB_+$ .

Task	Accuracy						
	$\{t_0, t_1\}$		$\{t_1, t_2\}$		$\{t_2, t_3\}$		
correctly detected parents	15/19	79%	20/21	95%	7/10	70%	
correctly detected children	35/38	92%	32/42	76%	14/20	70%	
correct parent–children triplets	15/19	78%	16/21	76%	7/10	70%	
correctly registered cell pairs	36/36	100%	44/49	90%	76/80	95%	

## 5. Conclusions and Future Work

We have developed a methodology for automatic cell tracking in recordings of dense bacterial colonies growing in a mono-layer. We have also validated our approach using synthetic data from agent based simulations, as well as experimental recordings of *E. coli* colonies growing in microfluidic traps. Our next goal is to streamline our implementation for systematic cell registration on experimentally acquired recordings of such cell colonies, to enable automated quantitative analysis and modeling of cell population dynamics and lineages.

There are a number of challenges for our cell tracking algorithm: Inherent imaging artifacts such as noise or intensity drifts, cell overlaps, similarity of cell shape characteristics across the population, tight packing of cells, somewhat large interframe times, cell growth combined with cell motion, and cell divisions represent just a few of these challenges. Overall, the cell tracking problem has combinatorial complexity, and for large frames is beyond the concrete patience of human experts. We tackle these challenges by developing a two-stage algorithm that first identifies parent–children triplets and subsequently computes cell registration from one frame to the next, after reducing the two original cell sets by automatic removal of the identified parent–children triplets. Our algorithms specify innovative cost functions dedicated to these registration challenges. These cost functions have combinatorial complexity. To discover good registrations, we minimize these cost functions numerically by intensive stochastic simulations of specifically structured BMs. We have validated the potential of our approach by reporting promising results obtained on long synthetic image sequences of simulated cell colonies (which naturally provide a ground truth for cell registration from one frame to the next). We have also successfully tested our algorithms on experimental recordings of live bacterial colonies.

The choice of adequate cost functions to drive each major cost optimization step in our multi-step cell tracking algorithms is essential for obtaining good tracking. Selecting the proper formulation had a strong impact on actual tracking accuracy. Our cost functions are fundamentally nonlinear, which entails additional complications. We introduced a set of meta-parameters for each cost function, and proposed an original learning algorithm to automatically identify good ranges for these meta-parameters.

Our BMs are focused on stochastic minimization of dedicated cost functions. An interesting feature of BMs we will explore in future work is the simplicity of their natural massive parallelization for fast stochastic minimization [90]. This allows us to mitigate the slow convergence typically observed for Gibbs samplers on discrete state spaces with high cardinality. Parallelized BMs implement a form of massively parallel simulated annealing. Sequential simulated annealing has been explored by physicists [105–108] seeking to minimize spin–glasses energies. For these clique-based energies, reaching global minima requires unfeasible CPU times, and much faster parallel simulated annealing yields only good local minima, via a sophisticated but still greedy stochastic search. Parallel stochasticity favors ending in rather stable local minima, which in turn enforces low sensitivity to small changes in energy parameters. Robustness to small changes in the coefficients of our cost functions is a desirable feature, since our algorithmic calibration of cost coefficients focuses on computing good ranges for these meta-parameters. We do not aim to seek global minima, generally a very elusive search because computing speed and scalability are important features in our problem. Recall the established results of Huber [109] showing that optimal estimators of the mean for a Gaussian distribution lose efficiency very quickly when the Gaussian data are slightly perturbed.

In future work, we will further improve the stability and accuracy of our cell registration algorithms by exploring natural modifications of our cost functions. In the present work, we have not yet explicitly considered the case of cells vanishing between successive frames. This is a critical issue that can occur due to cells exiting or entering the field of view as well as due to errors in cell segmentation. The problem is somewhat controlled and/or mitigated in our experimental setup, where we expect cells to enter or vanish close to a precisely positioned trap edge and/or near frame boundaries. Since we intend to

track lineages, each frame-to-frame error of this type may be problematic, and it will be instrumental for our future work to address these issues.

Linking parents to children involves an optimization distinct from the final optimization of frame-to-frame registrations. This did reduce computing time without reducing the quality for our benchmark results. However, in future work, one could attempt to iterate this sequence of two optimizations in order to reach a better minimum.

We note that our algorithm does work for experimental setups in which the frame rate of the video recordings is not fixed. This will require an adaptive parameter selection that depends on the frame rate. This can be implemented based on a trivial rescaling procedure. However, note that, for larger interframe times, more errors will impact tracking results. Indeed, large interframe durations intensify fluctuations in key parameters of cell dynamics, and increase the range of cell displacement, imposing searches in larger cell neighborhoods for cell pairing, as well as increased combinatorial complexity.

We have considered synthetic data to evaluate the performance of our method. One clear practical issue is that some of the parameters of our tracking algorithms may change when applied to laboratory image sequences acquired from colonies of different cells, with various image acquisition setups. One can design a computational framework to automatically fit the parameters of the simulation model to the imaging data acquired on specific live cell colonies, using specific camera hardware and setup. In future work, we will attempt to implement this type of fitting for our simulation model, before launching intensive model simulations to calibrate the parameters of our new tracking algorithms. We have not yet removed physical scales in the implementation of our tracking algorithm. Implementing such a non-dimensionalization will allow us to reduce the sensitivity of our methodology with respect to new datasets.

Identification of full lineages is an interesting concrete goal for cell tracking. Evaluating the accuracy of lineage identification on real cell colonies is quite challenging since it requires inheritable biological tagging of cells. This is probably feasible for populations mixing two or three cell types, but not for individualized tagging in populations of moderate size. However even partial tagging of sub-populations would provide some control on lineage identification accuracies.

**Author Contributions:** Conceptualization, R.A., M.R.B., K.J. and A.M.; methodology, R.A., A.M., S.S.; software, S.S. and J.J.W.; validation, S.S. and J.J.W.; formal analysis, R.A. and A.M.; investigation, R.A., M.R.B., K.J., A.M. and S.S.; resources, R.A., M.R.B., K.J. and A.M.; data curation, R.N.A. and M.R.B.; writing—original draft preparation, R.N.A., R.A., M.R.B., K.J., A.M., S.S. and J.J.W.; writing—review and editing, R.N.A., R.A., M.R.B., K.J., A.M., S.S. and J.J.W.; visualization, S.S., J.J.W. and A.M.; supervision, R.A. and A.M.; project administration, R.A., M.R.B., K.J. and A.M.; funding acquisition, R.A., M.R.B., K.J. and A.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partly supported by the National Science Foundation (NSF) through the grants GRFP 1842494 (R.N.A.), DMS-1854853 (R.A. and A.M.), DMS-2009923 (R.A. and A.M.), 1662305 (K.J.), MCB-1936770 (K.J.), and DMS-2012825 (A.M.); the joint NSF-National Institutes of General Medical Sciences Mathematical Biology Program grant DMS-1662290 (M.R.B.); and the Welch Foundation grant C-1729 (M.R.B.). Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF or the Welch Foundation.

**Acknowledgments:** This work was completed in part with resources provided by the Research Computing Data Core at the University of Houston.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A. Stochastic Dynamics of BMs

Notations and terminology refer to Section 3.4. Consider a BM network of  $N$  stochastic neurons  $U_j$ , with finite configuration set  $CONF = W(1) \times \dots \times W(N)$ . At time  $t$ , let  $Z_j(t) \in W(j)$  be the random state of neuron  $U_j$ , and the BM configuration  $Z(t) \in CONF$

is then  $Z(t) = \{Z_1(t), \dots, Z_N(t)\}$ . Fix as in Section 3.4 a sequence  $Temp(t)$  of virtual temperatures slowly decreasing to 0 for large  $t$ .

There are two main options to implement the Markov chain dynamics  $Z(t) \rightarrow Z(t+1)$  (see [95]).

#### Appendix A.1. Asynchronous BM Dynamics

Generate a long random sequence of sites  $m(t) \in S = \{1, \dots, N\}$ , for instance by concatenating successive random permutations of the set  $S$ . At time  $t$ , the only neuron that may modify its current state is  $U_{m(t)}$ . For brevity, write  $M = m(t)$ . The neuron  $U_M$  will compute its new random state  $Z_M(t+1) \in W(M)$  by the following *updating procedure*: (i) For each  $y$  in  $W(M)$ , define a new configuration  $Y \in \text{CONF}$  by  $Y_M(t) = y$ , and  $Y_j(t) = Z_j(t)$  for all  $j \neq M$ . Let  $\Delta(y) = E(Y) - E(Z(t))$  be the corresponding BM energy change. (ii) In the finite set  $W(M)$ , select any  $z$  such that  $\Delta(z) = \min_{y \in W(M)} \Delta(y)$ , and set  $D = \max\{0, \Delta(z)\}$ . (iii) Compute the probability  $p = \exp(-D/Temp(t))$ . (iv) The new random state  $Z_M(t+1)$  of neuron  $U_M$  will be equal to  $z$  with probability  $p$  and equal to the current state  $Z_M(t)$  with probability  $1 - p$ . (v) For all  $j \neq M$ , the new state  $Z_j(t+1)$  of neuron  $U_j$  remains equal to its current state  $U_j(t)$ .

#### Appendix A.2. Synchronous BM Dynamics

Fix a *synchrony* parameter  $0 < \alpha < 1$ , usually around 50%. At each time  $t$ , all neurons  $U_j$  synchronously, but independently compute their own random *binary tag*  $tag_j(t)$ , equal to 1 with probability  $\alpha$ , and to 0 with probability  $(1 - \alpha)$ . Let  $\text{SYN}(t)$  be the set of all neurons. All the neurons  $U_j$  such that  $tag_j(t) = 1$  then synchronously and independently compute their new random states  $Z_j(t+1) \in W(j)$  by applying the updating procedure given above. In addition, for all  $j$  such that  $tag_j(t) = 0$ , the new state  $Z_j(t+1)$  of  $U_j$  remains equal to  $Z_j(t)$ .

#### Appendix A.3. Comparing Asynchronous and Synchronous BM Dynamics

As  $t$  becomes large, and for temperatures  $Temp(t)$  slowly decreasing to 0, both BM dynamics generate with high probability configurations  $Z(t)$  which provide deep local minima  $E(Z(t))$  of the BM energy function. The asynchronous dynamics can be fairly slow. However, the synchronous dynamics are much faster since they emulate efficient forms of *parallel simulated annealing* (see [90,110]) and are directly implementable on GPUs.

### Appendix B. Computer Hardware

The computations were carried out on a dedicated server at the Department of Mathematics of the University of Houston. The hardware specifications are 64 Intel(R) Xeon(R) Gold 6142 CPU cores at 2.60 GHz with 128 GB of memory.

### Appendix C. Parameters for Simulation Software

Our tracking module is a collection of python functions and has been released to the public at <https://github.com/scopagroup/BacTrak> (accessed on 15 December 2021). We refer to [12,81] for a detailed description of this mathematical model and its implementation. The code for generating the synthetic data has been released at <https://github.com/jwinkle/eQ> (accessed on 15 December 2021). We note that detailed installation instructions for the software can be found on this page. The parameters for this agent-based simulation software are as follows: Cells were modeled as 2D spherocylinders of constant, 1  $\mu\text{m}$  width. The computational framework takes into account mechanical constraints that can impact cell growth and influence other aspects of cell behavior. The growth rate of the cells is exponential and is controlled by the doubling time. The time until cells double is set to 20 min (default setting; resulting in a growth rate of  $g.rate = 1.05$ ). The cells have a length of approximately 2  $\mu\text{m}$  after division and 4  $\mu\text{m}$  right before division (minimum division length of 4  $\mu\text{m}$ ; subject to some random perturbation). In our data set

of simulated videos, there is no “trap wall” (as opposed to the simulations carried out in [12,81]). The “trap” encompassing all cells on a given frame has a size of  $30\ \mu\text{m} \times 30\ \mu\text{m}$  subdivided into  $400 \times 400$  pixels of size  $0.075\ \mu\text{m} \times 0.075\ \mu\text{m}$ . The size of the resulting binary image used in our tracking algorithm is  $600 \times 600$  pixels. (We add a boundary of 100 pixels on each side). Bacteria are moving, growing and dividing within the trap. However, at this stage of our study, we consider only video segments where no cell disappears and where cells do not enter the trap from outside so that the trap is a confined environment. Cells move only due to soft shocks’ interactions with other neighboring cells. The time interval between any two successive image frames ranges from one minute to six minutes (see Table 1). All other simulation parameters remain unchanged; i.e., we use the default parameters specified in the simulation software.

## Appendix D. Cell Segmentation

In the next couple of sections, we outline the framework we have developed to segment individual cells from real world laboratory imaging data. In a first step, we consider traditional segmentation algorithms—a watershed algorithm [103,111,112] in combination with a variational contour based model—to generate a sufficiently large dataset to train a neuronal network. The actual segmentations on real data can subsequently be carried out efficiently using segmentation predictions generated by the trained neuronal network. Note that the proposed segmentation algorithm is only included for completeness. We do not view this as a major contribution of the present work.

### Appendix D.1. Watershed Algorithm

We consider a *watershed algorithm* based on immersion that compares high intensity values to local intensity minima for cell segmentation [103,111,112].

We consider Matlab’s implementation of the watershed algorithm in the present work. This version of the watershed algorithm is unseeded and yields  $n$  regions  $R = \{R_1, R_2, \dots, R_n\}$ . To identify these regions, we perform a statistical analysis of each image histogram to compute adaptive rough thresholds for interiors and exterior of cells. This leads to watershed results which identify each cell by a segment slightly larger than the cell itself. The very small percentage of oversegmented cells is automatically detected by cell length and width computations through PCA analysis of each cell shape viewed as a cloud of planar points. Since our segments are slightly too wide, we reduce each segment to the exact outer cell contour by applying a Mumford–Shah algorithm to each segment computed by the watershed algorithm. In an ideal case, after applying the watershed algorithm, each individual bacteria cell  $b_i$ ,  $i = 1, \dots, n$ , will be located in a single region  $R_i \subset \mathbb{R}^2$ . However, we observed several segmentation errors after applying the watershed algorithm to the considered data. A common error is that a line segment that defines the boundary of a region crosses through a cell. That is, two regions contain parts of one bacterium cell. In what follows, we devise strategies to correct these errors. For this processing step, we have normalized the intensities of the data to  $[0, 1]$ .

### Appendix D.2. Segmentation Errors: Correction Steps

We define the boundary segment  $B_{i,j}$  as a non-empty intersection of two region’s boundaries, i.e.,  $B_{i,j} = \partial S_i \cap \partial S_j$ . Moreover, we denote the area of a region  $R_i$  as  $\text{area}(R_i)$ . We know that the interior of a bacteria cell  $b_i$  has a lower intensity than the exterior region of a cell. More precisely, the interior of a cell tends to have intensity values of zero, whereas the exterior of a cell (i.e., the background) tends to have an intensity that is close or equal to one. For this reason, we define a function for the intensity of the boundary. To remove outliers, we consider the average intensity value of the pixels located along a boundary segment. We denote this mean intensity value along a boundary  $B_{i,j}$  by  $\text{mint}(B_{i,j})$  and the average intensity of a region  $R_i$  by  $\text{mint}(R_i)$ . One difficulty is that we cannot assume that the intensity of the pixels on the interior of each cell corresponds to the same value (i.e., there exist intensity and contrast drifts depending on location). We hypothesize that, if



$\text{mint}(B_{i,j})$  of a boundary segment is close to the average intensity of the regions on both sides of the boundary segment  $B_{i,j}$ , this boundary segment does not separate two bacteria cells; it is erroneous. Conversely, if the difference between the mean intensity along a boundary segment and the mean intensity of the interior regions it separates is high, we consider that the boundary segment represents a good segmentation (i.e., represents a segment that does separate two cells). To quantify this notion, we define the height of a boundary segment as  $H_{i,j} = \text{mint}(B_{i,j}) - (\text{mint}(R_i) + \text{mint}(R_j))/2$ .

In Table A1, we report some statistics associated with the quantities of interest introduced above. There are several key observations we can draw from this table which confirm our qualitative (i.e., visual) assessment of the segmentation results. Most notably, we can observe that there seem to exist outliers in terms of cell size. Moreover, we can observe that, in some cases, we obtain a height of the boundary segment that is negative, and by that nonsensical. These observations allow us to develop some heuristic rules to remove erroneous segmentations.

**Table A1.** Statistics of some quantities of interest related to the intensity of boundary segments and regions. These quantities allow us to define heuristics to identify erroneous segmentations computed by the watershed algorithm. We state the characteristic and report the minimum, maximum 5% quantile, mean, and standard deviation for the reported quantities of interest.

Characteristic	5% Quantile	Min	Max	Mean
Watershed area	56.00	43.00	984.00	$211.00 \pm 138.00$
Mean intensity of area	0.34	0.00	0.57	$0.41 \pm 0.06$
Mean intensity of boundary segment	0.46	0.30	0.99	$0.74 \pm 0.14$
Height of boundary segment	0.05	−0.09	0.62	$0.33 \pm 0.14$

We introduce the following post-processing steps: (i) We connect small regions to their neighbors (i.e., regions that are too small in area to realistically contain any cells). We select the threshold for the area to be 65. This threshold is selected in accordance with the scale of the image and the expected size of bacteria cells observed in the image data. We merge each small region with one of its neighboring regions by removing the segment that separates the two. To select an appropriate region for merging, we choose the region that gave the lowest height  $H_{i,j}$  from all available candidate regions that share the same boundary segment. (ii) We remove all boundary segments  $B_{i,j}$  with a height  $H_{i,j}$  that is below the 5% quantile of all heights. (iii) We remove all incomplete regions from our segmentation. We define a region as incomplete, if the region or the associated boundary segments touch an edge of the image. This step is necessary since we cannot guarantee that the regions close to the boundary contain an entire cell or only parts of a cell. Consequently, we decided to remove them to prevent any issues with our post-analysis.

### Appendix D.3. Cell Boundary Detection

The next step is to identify the boundaries of individual cells contained within a subregion defined by the watershed algorithm. To identify the boundaries of the cells (and by that segment the individual cells), we use the Mumford–Shah algorithm [104]. Notice that we can execute the Mumford–Shah algorithm for each region  $R_i$  separately making this an embarrassingly parallelizable problem. Denote the cell in each  $R_i$  region by  $b_i$ . We divide each of these regions into three different zones. The first zone is the interior of the cell  $b_i$  denoted by  $\text{in}(b_i)$ . The second zone is exterior of the cell (i.e., the background) contained in the region and denoted by  $\text{out}(b_i)$ . The third zone is the boundary of the cell  $b_i$ , denoted by  $\partial b_i$ . The Mumford–Shah algorithm represents a variational approach that allows us to segment cartoon like images. Mathematically speaking, we model information contained in each region  $R_i$  as piecewise-smooth functions. In our model, the associated regions we seek to identify are given by the zones defined above—the interior and the exterior of the cell  $b_i$ . Let  $u_{\text{int}}(b_i)$  denote the mean intensity for the interior of the cell  $b_i$  and

$u_{\text{ext}}(b_i)$  denote the mean intensity for the exterior of the cell  $b_i$ . With this definition, we obtain the cost functional

$$\text{cost}_{\text{MS}}(\text{int}(b_i), \text{ext}(b_i)) = \sum_{x \in \text{ext}(b_i)} (u(x) - u_{\text{ext}}(b_i))^2 + \sum_{x \in \text{int}(b_i)} (u(x) - u_{\text{int}}(b_i))^2 + \nu \text{bl}(b_i),$$

where the first two terms measure the discrepancy between the piecewise smooth function  $u_{\text{ext}}$  and  $u_{\text{int}}$  and the image intensities  $u$  and the third term is a penalty that measures the length of the boundary of a particular cell  $b_i$  with parameter  $\nu > 0$ . Notice that our formulation slightly deviates from the traditional definition of the Mumford–Shah cost functional; we drop the penalty for the smoothness of the function  $u$ . The minimizer of the cost function  $\text{cost}_{\text{MS}}$  defined above provides the sought after segmentation: the boundary, interior, and exterior of a cell. We have implemented the minimization of the cost function formula for each cell separately.

#### Appendix D.4. Convolutional Neural Networks (CNNs)

Next, we introduce our actual method for cell segmentation that can be efficiently applied to a large dataset (as opposed to the prototype method described above to generate the underlying training data). The biggest issue with the methodology outlined above is that our prototype implementation is computationally costly. While we envision that an improved implementation as well as the use of parallel computing can significantly reduce the time to solution, we decided not to further pursue a reduction in runtime but extend our methodology by taking advantage of existing machine learning algorithms. Replacing the approach outlined above by CNNs allowed us to reduce the runtime by factor of 60 to less than 3 min, without any significant loss in accuracy.

**Training and Testing Data.** In the absence of any ground truth data set for the classification of rod-shape bacteria cells from movies of cell populations, we consider the output of the Mumford–Shah algorithm introduced above as ground truth classification for training and testing our machine learning methodology. Above, we introduced three different zones: The interior  $\text{int}(b_i)$ , the exterior  $\text{ext}(b_i)$ , and the boundary  $\partial b_i$  of a cell  $b_i$ . We reduce these three regions to two zones—the interior and exterior of a cell  $b_i$ . We assign pixels that belong to  $\text{int}(b_i)$  the label 0 and pixels that belong to  $\text{ext}(b_i)$  and  $\partial b_i$  the label of 1. For an image of size  $200 \times 200$ , we obtain 40,000 binary labels. We limit the training of the CNN to a subregion of size  $200 \times 200$  in the center of each preprocessed image to avoid issues associated with mislabeled training data of cells located at the boundary of our data. We consider  $X$  as the set of features and  $Y$  as the set of labels. We want to assign to each pixel a label of either 0 or 1. For pixel  $p$ , we define  $X_p$  to be a  $7 \times 7$  square window with center  $p$  located in the original image. The corresponding label  $Y_p$  is denoted by  $C(p)$ , which corresponds to the class of the pixel  $p$  in the binarized image.

**CNN Algorithm.** The considered CNN algorithm consists of two parts, (i) the convolutional auto-encoder and (ii) a fully connected multilayer perceptron (MLP). The input for the auto-encoder is a window of  $7 \times 7$  pixels. In the first layer of the encoder, we have a  $5 \times 5 \times 4$  convolution layer Conv1 with  $3 \times 3$  kernel. We feed Conv1 to a max-pooling layer MPool2 with one stride and pooling window  $2 \times 2$ . The output of MPool2 is the input of a  $3 \times 3 \times 8$  convolution layer Conv3. For decoding, we have almost the same structure in reverse order: We feed Conv3 to a  $5 \times 5 \times 4$  deconvolution with  $3 \times 3$  kernel. Subsequently, we feed the output of this layer to a  $7 \times 7 \times 1$  deconvolution with  $3 \times 3$  kernel. The decoder's output is a window of  $7 \times 7$  pixels. We compare this output with the input window (since it is an auto-encoder, features and labels are the same) by using the mean square error as a cost function. We train the auto-encoder for all training sets using a mini-batch gradient descent. When the training is finished, we freeze the weights for Conv1 and Conv3.

After training the auto-encoder and freezing the weights, we feed  $X$  as the input to Conv1 and obtain the output of Conv3 denoted by  $\hat{X}$ . In the next step, we train an MLP with features  $\hat{X}$  and labels  $Y$ . We flatten  $\hat{X}$ , which is a  $3 \times 3 \times 8$  matrix to a vector of size

$72 \times 1$ , called FCL4. FCL4 is fully connected to the hidden layer HID5 with 10 nodes. We use ReLu as a nonlinear function for HID5. We connect HID5 to the output layer OUT6, which possess two nodes for the two classes 0 and 1. We use a softmax function to find two probabilistic outputs  $p_0$  and  $p_1 = 1 - p_0$  for related classes. We use maximum-entropy as a cost function. We train the MLP for training set of  $(\hat{X}, Y)$  with mini-batch gradient descent.

We have trained the model with two images of size  $200 \times 200$  pixels; the training set is 80,000  $7 \times 7$  images. We train the model for 100 epochs. The accuracy of the model for the image is 93%. The confusion matrix is shown in Table A2. Based on this confusion matrix, we can observe that the proposed methodology can predict the pixels located in the interior of a cell quite well. However, we can also observe that there is a slightly lower accuracy for the pixels outside the cells. This can be probably explained by the fact that the data sets are tightly packed with cells so that we have available more observations of foreground pixels (interior of cells) than pixels that belong to the background.

**Table A2.** Confusion matrix for the CNN.

	0	1
0	0.97	0.03
1	0.11	0.89

## References

- Butts-Wilmsmeyer, C.J.; Rapp, S.; Guthrie, B. The technological advancements that enabled the age of big data in the environmental sciences: A history and future directions. *Curr. Opin. Environ. Sci. Health* **2020**, *18*, 63–69. [\[CrossRef\]](#)
- Sivarajah, U.; Kamal, M.M.; Irani, Z.; Weerakkody, V. Critical analysis of Big Data challenges and analytical methods. *J. Bus. Res.* **2017**, *70*, 263–286. [\[CrossRef\]](#)
- Balomenos, A.D.; Tsakanikas, P.; Aspidou, Z.; Tampakaki, A.P.; Koutsoumanis, K.P.; Manolakis, E.S. Image analysis driven single-cell analytics for systems microbiology. *BMC Syst. Biol.* **2017**, *11*, 1–21. [\[CrossRef\]](#)
- Klein, J.; Leupold, S.; Biegler, I.; Biedendieck, R.; Münch, R.; Jahn, D. TLM-Tracker: Software for cell segmentation, tracking and lineage analysis in time-lapse microscopy movies. *Bioinformatics* **2012**, *28*, 2276–2277. [\[CrossRef\]](#)
- Stylianidou, S.; Brennan, C.; Nissen, S.B.; Kuwada, N.J.; Wiggins, P.A. SuperSegger: Robust image segmentation, analysis and lineage tracking of bacterial cells. *Mol. Microbiol.* **2016**, *102*, 690–700. [\[CrossRef\]](#) [\[PubMed\]](#)
- Bennett, M.R.; Hasty, J. Microfluidic devices for measuring gene network dynamics in single cells. *Nat. Rev. Genet.* **2009**, *10*, 628–638. [\[CrossRef\]](#)
- Danino, T.; Mondragón-Palomino, O.; Tsimring, L.; Hasty, J. A synchronized quorum of genetic clocks. *Nature* **2010**, *463*, 326–330. Available online: <http://xxx.lanl.gov/abs/15334406> (accessed on 15 December 2021). [\[CrossRef\]](#)
- Mather, W.; Mondragon-Palomino, O.; Danino, T.; Hasty, J.; Tsimring, L.S. Streaming instability in growing cell populations. *Phys. Rev. Lett.* **2010**, *104*, 208101. [\[CrossRef\]](#) [\[PubMed\]](#)
- El Najjar, N.; Van Teeseling, M.C.; Mayer, B.; Hermann, S.; Thanbichler, M.; Graumann, P.L. Bacterial cell growth is arrested by violet and blue, but not yellow light excitation during fluorescence microscopy. *BMC Mol. Cell Biol.* **2020**, *21*, 35. [\[CrossRef\]](#)
- Icha, J.; Weber, M.; Waters, J.C.; Norden, C. Phototoxicity in live fluorescence microscopy, and how to avoid it. *BioEssays* **2017**, *39*, 1700003. [\[CrossRef\]](#) [\[PubMed\]](#)
- Kim, J.K.; Chen, Y.; Hirning, A.J.; Alnahhas, R.N.; Josić, K.; Bennett, M.R. Long-range spatio-temporal coordination of gene expression in synthetic microbial consortia. *Nat. Chem. Biol.* **2019**, *15*, 1102–1109. [\[CrossRef\]](#)
- Winkle, J.; Igoshin, O.A.; Bennett, M.R.; Josic, K.; Ott, W. Modeling mechanical interactions in growing populations of rod-shaped bacteria. *Phys. Biol.* **2017**, *14*, 055001. [\[CrossRef\]](#) [\[PubMed\]](#)
- Carpenter, A.E.; Jones, T.R.; Lamprecht, M.R.; Clarke, C.; Kang, I.H.; Friman, O.; Guertin, D.A.; Chang, J.H.; Lindquist, R.A.; Moffat, J.; et al. CellProfiler: Image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **2006**, *7*, R100. [\[CrossRef\]](#) [\[PubMed\]](#)
- Kamentsky, L.; Jones, T.R.; Fraser, A.; Bray, M.; Logan, D.; Madden, K.; Ljosa, V.; Rueden, C.; Harris, G.B.; Eliceiri, K.; et al. Improved structure, function, and compatibility for CellProfiler: modular high-throughput image analysis software. *Bioinformatics* **2011**, *27*, 1179–1180. [\[CrossRef\]](#)
- McQuin, C.; Goodman, A.; Chernyshev, V.; Kamentsky, L.; Cimini, B.A.; Karhohs, K.W.; Doan, M.; Ding, L.; Rafelski, S.M.; Thirstrup, D.; et al. CellProfiler 3.0: Next-generation image processing for biology. *PLoS Biol.* **2018**, *16*, e2005970. [\[CrossRef\]](#)
- Alnahhas, R.N.; Sadeghpour, M.; Chen, Y.; Frey, A.A.; Ott, W.; Josić, K.; Bennett, M.R. Majority sensing in synthetic microbial consortia. *Nat. Commun.* **2020**, *11*, 1–10. [\[CrossRef\]](#) [\[PubMed\]](#)
- Locke, J.C.W.; Elowitz, M.B. Using movies to analyse gene circuit dynamics in single cells. *Nat. Rev. Microbiol.* **2009**, *7*, 383–392. [\[CrossRef\]](#)

18. Alnahhas, R.N.; Winkle, J.J.; Hirning, A.J.; Karamched, B.; Ott, W.; Josić, K.; Bennett, M.R. Spatiotemporal Dynamics of Synthetic Microbial Consortia in Microfluidic Devices. *ACS Synth. Biol.* **2019**, *8*, 2051–2058. [[CrossRef](#)] [[PubMed](#)]
19. Hand, A.J.; Sun, T.; Barber, D.C.; Hose, D.R.; MacNeil, S. Automated tracking of migrating cells in phase-contrast video microscopy sequences using image registration. *J. Microsc.* **2009**, *234*, 62–79. [[CrossRef](#)] [[PubMed](#)]
20. Ulman, V.; Maška, M.; Magnusson, K.E.G.; Ronneberger, O.; Haubold, C.; Harder, N.; Matula, P.; Matula, P.; Svoboda, D.; Radojevic, M.; et al. An objective comparison of cell-tracking algorithms. *Nat. Methods* **2017**, *14*, 1141–1152. [[CrossRef](#)] [[PubMed](#)]
21. Marvasti-Zadeh, S.M.; Cheng, L.; Ghanei-Yakhdan, H.; Kasaei, S. Deep learning for visual tracking: A comprehensive survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, 1–26. [[CrossRef](#)]
22. Yilmaz, A.; Javed, O.; Shah, M. Object tracking: A survey. *ACM Comput. Surv. (CSUR)* **2006**, *38*, 13-es. [[CrossRef](#)]
23. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the International Conference on Artificial Intelligence, Vancouver, BC, Canada, 24–28 August 1981; pp. 674–679.
24. Mang, A.; Biros, G. An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration. *SIAM J. Imaging Sci.* **2015**, *8*, 1030–1069. [[CrossRef](#)] [[PubMed](#)]
25. Mang, A.; Ruthotto, L. A Lagrangian Gauss–Newton–Krylov solver for mass- and intensity-preserving diffeomorphic image registration. *SIAM J. Sci. Comput.* **2017**, *39*, B860–B885. [[CrossRef](#)] [[PubMed](#)]
26. Mang, A.; Gholami, A.; Davatzikos, C.; Biros, G. CLAIRE: A distributed-memory solver for constrained large deformation diffeomorphic image registration. *SIAM J. Sci. Comput.* **2019**, *41*, C548–C584. [[PubMed](#)]
27. Borzi, A.; Ito, K.; Kunisch, K. An optimal control approach to optical flow computation. *Int. J. Numer. Methods Fluids* **2002**, *40*, 231–240. [[CrossRef](#)]
28. Horn, B.K.P.; Shunck, B.G. Determining optical flow. *Artif. Intell.* **1981**, *17*, 185–203. [[CrossRef](#)]
29. Delpiano, J.; Jara, J.; Scheer, J.; Ramírez, O.A.; Ruiz-del Solar, J.; Härtel, S. Performance of optical flow techniques for motion analysis of fluorescent point signals in confocal microscopy. *Mach. Vis. Appl.* **2012**, *23*, 675–689.
30. Madrigal, F.; Hayet, J.B.; Rivera, M. Motion priors for multiple target visual tracking. *Mach. Vis. Appl.* **2015**, *26*, 141–160. [[CrossRef](#)]
31. Banerjee, D.S.; Stephenson, G.; Das, S.G. Segmentation and analysis of mother machine data: SAM. *bioRxiv* **2020**. [[CrossRef](#)]
32. Jug, F.; Pietzsch, T.; Kainmüller, D.; Funke, J.; Kaiser, M.; van Nimwegen, E.; Rother, C.; Myers, G. Optimal Joint Segmentation and Tracking of Escherichia Coli in the Mother Machine. In *Bayesian and Graphical Models for Biomedical Imaging*; Springer: Cham, Switzerland, 2014; Volume LNCS 8677, pp. 25–36.
33. Lugagne, J.B.; Lin, H.; Dunlop, M.J. DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. *PLoS Comput. Biol.* **2020**, *16*, e1007673. [[CrossRef](#)] [[PubMed](#)]
34. Ollion, J.; Elez, M.; Robert, L. High-throughput detection and tracking of cells and intracellular spots in mother machine experiments. *Nat. Protoc.* **2019**, *14*, 3144–3161. [[CrossRef](#)]
35. Sauls, J.T.; Schroeder, J.W.; Brown, S.D.; Le Treut, G.; Si, F.; Li, D.; Wang, J.D.; Jun, S. Mother machine image analysis with MM3. *bioRxiv* **2019**, 810036. [[CrossRef](#)]
36. Smith, A.; Metz, J.; Pagliara, S. MMHelper: An automated framework for the analysis of microscopy images acquired with the mother machine. *Sci. Rep.* **2019**, *9*, 10123.
37. Arbelle, A.; Reyes, J.; Chen, J.Y.; Lahav, G.; Raviv, T.R. A probabilistic approach to joint cell tracking and segmentation in high-throughput microscopy videos. *Med. Image Anal.* **2018**, *47*, 140–152.
38. Okuma, K.; Taleghani, A.; De Freitas, N.; Little, J.J.; Lowe, D.G. A boosted particle filter: Multitarget detection and tracking. In Proceedings of the European Conference on Computer Vision, Prague, Czech Republic, 11–14 May 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 28–39.
39. Smal, I.; Niessen, W.; Meijering, E. Bayesian tracking for fluorescence microscopic imaging. In Proceedings of the 3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro, Arlington, VA, USA, 6–9 April 2006; pp. 550–553.
40. Kervrann, C.; Trubuil, A. Optimal level curves and global minimizers of cost functionals in image segmentation. *J. Math. Imaging Vis.* **2002**, *17*, 153–174. [[CrossRef](#)]
41. Li, K.; Miller, E.D.; Chen, M.; Kanade, T.; Weiss, L.E.; Campbell, P.G. Cell population tracking and lineage construction with spatiotemporal context. *Med. Image Anal.* **2008**, *12*, 546–566. [[CrossRef](#)] [[PubMed](#)]
42. Wang, X.; He, W.; Metaxas, D.; Mathew, R.; White, E. Cell segmentation and tracking using texture-adaptive snakes. In Proceedings of the IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Arlington, VA, USA, 12–15 April 2007; pp. 101–104.
43. Yang, F.; Mackey, M.A.; Ianzini, F.; Gallardo, G.; Sonka, M. Cell segmentation, tracking, and mitosis detection using temporal context. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Palm Springs, CA, USA, 26–29 October 2005; pp. 302–309.
44. Sethuraman, V.; French, A.; Wells, D.; Kenobi, K.; Pridmore, T. Tissue-level segmentation and tracking of cells in growing plant roots. *Mach. Vis. Appl.* **2012**, *23*, 639–658. [[CrossRef](#)]
45. Balomenos, A.D.; Tsakanikas, P.; Manolakis, E.S. Tracking single-cells in overcrowded bacterial colonies. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milano, Italy, 25–29 August 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 6473–6476.

46. Bise, R.; Yin, Z.; Kanade, T. Reliable cell tracking by global data association. In Proceedings of the IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Chicago, IL, USA, 30 March–2 April 2011; pp. 1004–1010.
47. Bise, R.; Li, K.; Eom, S.; Kanade, T. Reliably tracking partially overlapping neural stem cells in DIC microscopy image sequences. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention Workshop, London, UK, 20–24 September 2009; pp. 67–77.
48. Kanade, T.; Yin, Z.; Bise, R.; Huh, S.; Eom, S.; Sandbothe, M.F.; Chen, M. Cell image analysis: Algorithms, system and applications. In Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision (WACV), Kona, HI, USA, 5–7 January 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 374–381.
49. Primet, M.; Demarez, A.; Taddei, F.; Lindner, A.; Moisan, L. Tracking of cells in a sequence of images using a low-dimensional image representation. In Proceedings of the IEEE International Symposium on Biomedical Imaging, Paris, France, 14–17 May 2008; pp. 995–998.
50. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer Assisted Intervention, Munich, Germany, 5–9 October 2015; Volume LNCS 9351, pp. 234–241.
51. Su, H.; Yin, Z.; Huh, S.; Kanade, T. Cell segmentation in phase contrast microscopy images via semi-supervised classification over optics-related features. *Med. Image Anal.* **2013**, *17*, 746–765. [[CrossRef](#)]
52. Wang, Q.; Niemi, J.; Tan, C.M.; You, L.; West, M. Image segmentation and dynamic lineage analysis in single-cell fluorescence microscopy. *Cytom. Part A J. Int. Soc. Adv. Cytom.* **2010**, *77*, 101–110. [[CrossRef](#)] [[PubMed](#)]
53. Jiuqing, W.; Xu, C.; Xianhang, Z. Cell tracking via structured prediction and learning. *Mach. Vis. Appl.* **2017**, *28*, 859–874. [[CrossRef](#)]
54. Zhou, Z.; Wang, F.; Xi, W.; Chen, H.; Gao, P.; He, C. Joint multi-frame detection and segmentation for multi-cell tracking. In Proceedings of the International Conference on Image and Graphics, Beijing, China, 23–25 August 2019; Volume LNCS 11902, pp. 435–446.
55. Sixta, T.; Cao, J.; Seebach, J.; Schnittler, H.; Flach, B. Coupling cell detection and tracking by temporal feedback. *Mach. Vis. Appl.* **2020**, *31*, 1–18.
56. Hayashida, J.; Nishimura, K.; Bise, R. MPM: Joint representation of motion and position map for cell tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 3823–3832.
57. Payer, C.; Stern, D.; Neff, T.; Bishof, H.; Urschler, M. Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks. In Proceedings of the Medical Image Computing and Computer Assisted Intervention, Granada, Spain, 16–20 September 2018; Volume LNCS 11071, pp. 3–11.
58. Payer, C.; Štern, D.; Feiner, M.; Bischof, H.; Urschler, M. Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks. *Med. Image Anal.* **2019**, *57*, 106–119. [[PubMed](#)]
59. Vicar, T.; Balvan, J.; Jaros, J.; Jug, F.; Kolar, R.; Masarik, M.; Gumulec, J. Cell segmentation methods for label-free contrast microscopy: Review and comprehensive comparison. *BMC Bioinform.* **2019**, *20*, 1–25.
60. Al-Kofahi, Y.; Zaltsman, A.; Graves, R.; Marshall, W.; Rusu, M. A deep learning-based algorithm for 2D cell segmentation in microscopy images. *BMC Bioinform.* **2018**, *19*, 1–11.
61. Falk, T.; Mai, D.; Bensch, R.; Çiçek, Ö.; Abdulkadir, A.; Marrakchi, Y.; Böhm, A.; Deubner, J.; Jäckel, Z.; Seiwald, K.; et al. U-Net: Deep learning for cell counting, detection, and morphometry. *Nat. Methods* **2019**, *16*, 67–70. [[CrossRef](#)]
62. Lux, F.; Matula, P. DIC image segmentation of dense cell populations by combining deep learning and watershed. In Proceedings of the 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), Venice, Italy, 8–11 April 2019; pp. 236–239.
63. Moen, E.; Bannon, D.; Kudo, T.; Graf, W.; Covert, M.; Van Valen, D. Deep learning for cellular image analysis. *Nat. Methods* **2019**, *16*, 1233–1246. [[PubMed](#)]
64. Rempfler, M.; Stierle, V.; Ditzel, K.; Kumar, S.; Paulitschke, P.; Andres, B.; Menze, B.H. Tracing cell lineages in videos of lens-free microscopy. *Med. Image Anal.* **2018**, *48*, 147–161.
65. Stringer, C.; Wang, T.; Michaelos, M.; Pachitariu, M. Cellpose: A generalist algorithm for cellular segmentation. *Nat. Methods* **2021**, *18*, 100–106. [[CrossRef](#)]
66. Akram, S.U.; Kannala, J.; Eklund, L.; Heikkilä, J. Joint cell segmentation and tracking using cell proposals. In Proceedings of the IEEE 13th International Symposium on Biomedical Imaging (ISBI), Prague, Czech Republic, 13–16 April 2016; pp. 920–924.
67. Nishimura, K.; Hayashida, J.; Wang, C.; Bise, R. Weakly-Supervised Cell Tracking via Backward-and-Forward Propagation. In Proceedings of the European Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 104–121.
68. Rempfler, M.; Kumar, S.; Stierle, V.; Paulitschke, P.; Andres, B.; Menze, B.H. Cell lineage tracing in lens-free microscopy videos. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Quebec City, QC, Canada, 11–13 September 2017; pp. 3–11.
69. Maska, M.; Ulman, V.; Svoboda, D.; Matula, P.; Ederra, C.; Urbola, A.; Espana, T.; Venkatesan, S.; Balak, D.M.W.; et al. A benchmark for comparison of cell tracking algorithms. *Bioinformatics* **2014**, *30*, 1609–1617.
70. Löffler, K.; Scherr, T.; Mikut, R. A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction. *bioRxiv* **2021**, *16*, e0249257.
71. Vo, B.T.; Vo, B.N.; Cantoni, A. The cardinality balanced multi-target multi-Bernoulli filter and its implementations. *IEEE Trans. Signal Process.* **2008**, *57*, 409–423.



72. Pierskalla, W.P. The multidimensional assignment problem. *Oper. Res.* **1968**, *16*, 422–431. [\[CrossRef\]](#)
73. Gilbert, K.C.; Hofstra, R.B. Multidimensional assignment problems. *Decis. Sci.* **1988**, *19*, 306–321. [\[CrossRef\]](#)
74. Chakraborty, A.; Roy-Chowdhury, A.K. Context aware spatio-temporal cell tracking in densely packed multilayer tissues. *Med. Image Anal.* **2015**, *19*, 149–163. [\[CrossRef\]](#) [\[PubMed\]](#)
75. Liu, M.; Yadav, R.K.; Roy-Chowdhury, A.; Reddy, G.V. Automated tracking of stem cell lineages of Arabidopsis shoot apex using local graph matching. *Plant J.* **2010**, *62*, 135–147. [\[CrossRef\]](#)
76. Liu, M.; Chakraborty, A.; Singh, D.; Yadav, R.K.; Meenakshisundaram, G.; Reddy, G.V.; Roy-Chowdhury, A. Adaptive cell segmentation and tracking for volumetric confocal microscopy images of a developing plant meristem. *Mol. Plant* **2011**, *4*, 922–931. [\[CrossRef\]](#)
77. Liu, M.; Li, J.; Qian, W. A multi-seed dynamic local graph matching model for tracking of densely packed cells across unregistered microscopy image sequences. *Mach. Vis. Appl.* **2018**, *29*, 1237–1247. [\[CrossRef\]](#)
78. Vo, B.N.; Vo, B.T. A multi-scan labeled random finite set model for multi-object state estimation. *IEEE Trans. Signal Process.* **2019**, *67*, 4948–4963. [\[CrossRef\]](#)
79. Punchihewa, Y.G.; Vo, B.T.; Vo, B.N.; Kim, D.Y. Multiple object tracking in unknown backgrounds with labeled random finite sets. *IEEE Trans. Signal Process.* **2018**, *66*, 3040–3055. [\[CrossRef\]](#)
80. Kim, D.Y.; Vo, B.N.; Thian, A.; Choi, Y.S. A generalized labeled multi-Bernoulli tracker for time lapse cell migration. In Proceedings of the 2017 International Conference on Control, Automation and Information Sciences, Jeju, Korea, 18–21 October 2017; pp. 20–25.
81. Winkle, J.J.; Karamched, B.R.; Bennett, M.R.; Ott, W.; Josić, K. Emergent spatiotemporal population dynamics with cell-length control of synthetic microbial consortia. *PLoS Comput. Biol.* **2021**, *17*, e1009381.
82. Bise, R.; Sato, Y. Cell detection from redundant candidate regions under non-overlapping constraints. *IEEE Trans. Med Imaging* **2015**, *34*, 1417–1427. [\[CrossRef\]](#) [\[PubMed\]](#)
83. Matula, P.; Maška, M.; Sorokin, D.V.; Matula, P.; Ortiz-de Solórzano, C.; Kozubek, M. Cell tracking accuracy measurement based on comparison of acyclic oriented graphs. *PLoS ONE* **2015**, *10*, e0144959.
84. Agrawal, A.; Verschueren, R.; Diamond, S.; Boyd, S. A rewriting system for convex optimization problems. *J. Control Decis.* **2018**, *5*, 42–60. [\[CrossRef\]](#)
85. Diamond, S.; Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.* **2016**, *17*, 1–5.
86. Shen, X.; Diamond, S.; Gu, Y.; Boyd, S. Disciplined convex-concave programming. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 1009–1014.
87. Stricker, J.; Cookson, S.; Bennett, M.R.; Mather, W.H.; Tsimring, L.S.; Hasty, J. A fast, robust and tunable synthetic gene oscillator. *Nature* **2008**, *456*, 516–519. [\[CrossRef\]](#)
88. Chen, Y.; Kim, J.K.; Hirning, A.J.; Josić, K.; Bennett, M.R. Emergent genetic oscillations in a synthetic microbial consortium. *Science* **2015**, *349*, 986–989. [\[CrossRef\]](#)
89. Sloan, S.W. A fast algorithm for constructing Delaunay triangulations in the plane. *Adv. Eng. Softw.* **1987**, *9*, 34–55. [\[CrossRef\]](#)
90. Azencott, R. *Simulated Annealing: Parallelization Techniques*; Wiley-Interscience: Hoboken, NJ, USA, 1992; Volume 27.
91. Azencott, R.; Chalmond, B.; Coldefy, F. Markov Image Fusion to Detect Intensity Valleys. *Int. J. Comput. Vis.* **1994**, *16*, 135–145. [\[CrossRef\]](#)
92. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
93. Ackley, D.H.; Hinton, G.E.; Sejnowski, T.J. A learning algorithm for Boltzmann machines. *Cogn. Sci.* **1985**, *9*, 147–169. [\[CrossRef\]](#)
94. Hinton, G.E.; Sejnowski, T.J. Chapter Learning and Rerelearning in Boltzmann Machines. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*; MIT Press: Cambridge, MA, USA, 1986; pp. 282–317.
95. Azencott, R. Synchronous Boltzmann machines and Gibbs fields: Learning algorithms. In *Neurocomputing*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 51–63.
96. Azencott, R. Synchronous Boltzmann machines and artificial vision. *Neural Netw.* **1990**, 135–143. Available online: [https://www.math.uh.edu/~razencot/MyWeb/Research/Selected\\_Reprints/1990SynchronousBoltzmanMachinesArtificialVision.pdf](https://www.math.uh.edu/~razencot/MyWeb/Research/Selected_Reprints/1990SynchronousBoltzmanMachinesArtificialVision.pdf) (accessed on 15 December 2021).
97. Azencott, R.; Graffigne, C.; Labourdette, C. Edge Detection and Segmentation of Textured Plane Images. In *Stochastic Models, Statistical Methods, and Algorithms in Image Analysis*; Springer-Verlag: New York, NY, USA, 1992; Volume 74, pp. 75–88.
98. Kong, A.; Azencott, R. Binary Markov Random Fields and Interpretable Mass Spectra Discrimination. *Stat. Appl. Genet. Mol. Biol.* **2017**, *16*, 13–30. [\[CrossRef\]](#) [\[PubMed\]](#)
99. Azencott, R.; Doutriaux, A.; Younes, L. Synchronous Boltzmann Machines and Curve Identification Tasks. *Netw. Comput. Neural Syst.* **1993**, *4*, 461–480.
100. Garda, P.; Belhaire, E. An Analog Circuit with Digital I/O for Synchronous Boltzmann Machines. In *VLSI for Artificial Intelligence and Neural Networks*; Springer: Berlin, Germany, 1991; pp. 245–254.
101. Lafargue, V.; Belhaire, E.; Pujol, H.; Berechet, I.; Garda, P. Programmable Mixed Implementation of the Boltzmann Machine. In *International Conference on Artificial Neural Networks*; Springer: Berlin, Germany, 1994; pp. 409–412.

102. Pujol, H.; Klein, J.-O.; Belhaire, E.; Garda, P. RA: An analog neurocomputer for the synchronous Boltzmann machine. In Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, Turin, Italy, 26–28 September 1994; IEEE: Piscataway, NJ, USA, 1994; pp. 449–455.
103. Beucher, S.; Lantuejoul, C. Use of watersheds in contour detection. In *Workshop on Image Processing*; CCETT/IRISA: Rennes, France, 1979.
104. Mumford, D.B.; Shah, J. Optimal approximations by piecewise smooth functions and associated variational problems. *Commun. Pure Appl. Math.* **1989**, *42*, 577–685. [[CrossRef](#)]
105. Mézard, M.; Parisi, G.; Virasoro, M.A. *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications*; World Scientific Publishing Company: Singapore, 1987; Volume 9.
106. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [[CrossRef](#)] [[PubMed](#)]
107. Roussel-Ragot, P.; Dreyfus, G. A problem independent parallel implementation of simulated annealing: Models and experiments. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **1990**, *9*, 827–835. [[CrossRef](#)]
108. Burda, Z.; Krzywicki, A.; Martin, O.C.; Tabor, Z. From simple to complex networks: Inherent structures, barriers, and valleys in the context of spin glasses. *Phys. Rev. E* **2006**, *73*, 036110. [[CrossRef](#)] [[PubMed](#)]
109. Huber, P.J. The 1972 Wald Lecture Robust Statistics: A Review. *Ann. Math. Stat.* **1972**, *43*, 1041–1067. [[CrossRef](#)]
110. Ram, D.J.; Sreenivas, T.; Subramaniam, K.G. Parallel simulated annealing algorithms. *J. Parallel Distrib. Comput.* **1996**, *37*, 207–212. [[CrossRef](#)]
111. Digabel, H.; Lantuejoul, C. Iterative Algorithms. In Proceedings of the 2nd European Symposium Quantitative Analysis of Microstructures in Material Science, Biology and Medicine, Caen, France, 4–7 October 1977; pp. 85–89.
112. Vincent, L.; Soille, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991**, *13*, 583–598. [[CrossRef](#)]