Generator Platform of Benchmark Time-Lapsed Images Development of Cell Tracking Algorithms

Implementation of new features towards a realistic simulation of the cell spatial and temporal organization

Leonardo Martins[1], Pedro Canelas[1], André Mora[1], Andre S. Ribeiro[2] and José Fonseca[1]

[1]Computational Intelligence Group of CTS/UNINOVA. Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, Quinta da Torre, 2829-516, Caparica, Portugal
[2]Laboratory of Biosystem Dynamics, Dep. of Signal Processing, Tampere University of Technology, Tampere, Finland
l.martins@campus.fct.unl.pt, p.canelas@campus.fct.unl.pt, atm@uninova.pt, andre.ribeiro@tut.fi, jmf@uninova.pt

Abstract. Recent developments in live-cell microscopy imaging have led to the emergence of Single Cell Biology. This field has also been supported by the development of cell segmentation and tracking algorithms for data extraction. The validation of these algorithms requires benchmark databases, with manually labeled or artifi-cially generated images, so that the ground truth is known.. To generate realistic artificial images, we have developed a simulation platform capable of generating biologically inspired objects with various shapes and size, which are able to grow, divide, move and form specific clusters. Using this platform, we compared four tracking algorithms: Simple Nearest-Neighbor (NN), NN with Morphology (NNm) and two DBSCAN-based methodologies. We show that Simple NN per-forms well on objects with small velocities, while the others perform better for higher velocities and when objects form clusters. This platform for benchmark images generation and image analysis algorithms testing is openly available at (http://griduni.uninova.pt/Clustergen/ClusterGen _v1.0.zip).

Keywords: Microscopy, Synthetic Time-lapse Image Simulation, Cell Tracking, Cluster Tracking

1 INTRODUCTION

Recent advances in live-cell microscopy imaging have enhanced images quality and triggered the development of techniques for detecting and observing cellular structures and their kinetics [1, 2].

Live-cell imaging entails the tasks before and during image acquisition at the microscope, which includes image refinement, such as tuning illumination, focus, drift cor-

rection, stage positioning and microscope components selection (e.g. shutter, lens, camera, stage) [3]. It follows image processing (e.g., registration, segmentation, tracking, statistical quantification and background correction) [3, 4].

Microscopy software packages include automatic correction algorithms for noise attenuation, contrast correction, illumination compensation, etc [5].

The first step is usually image registration (overlay of two or more images of the same object at different instants, viewpoints or sensors) is a classical step with several methods available, which are based on modality, intensity, type of data, dimensionality, domain and type of transformation, and registration methodologies [6, 7].

The next step is the segmentation of cells or cellular structures of interest [8], where these segmented objects are detected, located, and separated from the background. The main challenge here is to automatize it with high specificity and sensitivity for a wide number of cases. Presently, there are various approaches, such as intensity thresholding, feature detection, morphological filtering, region accumulation, deformable model fitting, etc. [8].

When handling a time series, one needs track the objects between frames, i.e., to link the segmented objects in the actual frame with the ones from the previous frame, so as to attain the object's trajectory. With the information describing the target defined by the state sequence X_k, $k \epsilon \mathbb{N}$ (where $\mathbb{N}$ is the set of frames), and the measurements defined by Z_k, the goal of tracking is to estimate X_k, given all measurements until the moment $Z_{1:k}$ [9]. This is made difficult by noise, occlusions, illumination changes, complex motions and object's shape dynamics, which can enhance the misidentification of object tracks[10].

Currently available tools for tracking in different microscopy settings include the 'Cell-C', based on DAPI staining and fluorescence *in situ* hybridization images [11], 'CellTracer', which applies morphological methods to automatically segment bacterial cells, yeast and human cells [12], 'MicrobeTracker' and its accessory tool 'SpotFinder', which segment *Escherichia coli* and *Caulobacter crescentus* cells and detected fluorescent spots within [13], 'Schnitzcells', which segments and tracks *E. coli* cells in confocal or phase contrast images [14] and, 'CellAging', which was developed for cell segmentation and tracking in order to study the segregation and partitioning in cell division of protein aggregates [15].

The validation of these tools requires gold-standard images, usually manually annotated by biology experts. However, this validation is problematic, as it is expert-dependent (both inter-user and intra-user variability can be high) and is impractical in high-throughput data-sets [16]. To overcome this problem, a viable alternative is using artificial images of biologically inspired objects. These images, whose ground truth is known, can be used for the accurate quantitative evaluation of the image processing algorithms [4].

Next, we provide a comprehensive literature review of existing tools for simulation of synthetic microscopy images and of recent developments on cell tracking algorithms. In Section 3 we present the contributions to the development of the image simulation tools (models and parameters) and the implementation of three different tracking algorithms. In Section 4, the tracking results of several examples are presented using dif-

ferent parameters. Finally, in Section 5, we present our final remarks on the development of simulation tool and the results of the three algorithms, along with a description of potential future endeavors.

2 STATE OF THE ART

2.1 Synthetic Image Generators

There have been several contests and open challenges on microscopy image processing, usually requiring that each methodology is tested on the same benchmark data-sets (acquired by an independent laboratory or created by artificial image generators) [8]. Such artificial image generators require realistic biological models, and commonly use theoretical and experimental information on the statistical distributions of the object's behavior [17] and spatial and temporal data [18, 19]. If the object studied is a cell, these models should include morphology parameters such as cell shape and size, location of subcellular structures, kinetic and spatial statistics of cell growth, cell division, cell migration and models of internal cell functions.

The architecture of microscopy image simulators based on biological models can be divided in three main stages: the digital phantom object generation, the simulation of the signal passing through the optical system and the simulation of the image formed on a specific sensor [20].

Simulators such as 'SIMCEP' [21] have provided a gold-standard platform to validate and test image processing tools, such as the previously mentioned 'CellC' [11], the open-source and Java-based image processor ImageJ, and the commercially available MCID Analysis (Imaging Research Inc., Catharines, ON, Canada; Evaluation ver. 7.0), along with other image processing tools [22]. The phantom objects are generated with different cell parameters, such as probability of clustering, cell radius, and cell shape and with parameters related to the sensors and the optical system, such as background noise and illumination disturbance [22, 23].

'CytoPacq' is another toolbox specifically developed to simulate all three phases. For that, it is equipped with three different modules. The first module ('3D-cytogen') generates the digital object phantom, which imitates the cell structure and behavior generating microspheres, granulocytes, HL-60 Nucleus and images of Colon Tissue. The second module ('3D-optigen') simulates the transmission of the signal through the lenses, objective, excitation filter and emission filter (various sets of equipment can be simulated). The last module, '3D-acquigen' is the digital CCD camera simulator of the image capture process (noise, sampling, digitization) by changing the camera selection, the acquisition time, the dynamic range usage and the stage z-step [20, 24]. The same group also introduced a novel versatile tool ('TRAgen'), capable of generating 2D time-lapses by simulating live cell populations as a ground-truth for the evaluation of cell tracking algorithms. In this work, they included models of cell motility, division and clustering up to tissue-level density [25]. Both simulators have been an important step in the simulation of cellular dynamics, such as intracellular protein or RNA levels or even cell migration, division and growth [2, 3].

Another toolbox, called 'SimuCell' [26], is capable of generating artificial microscopy images with heterogeneous cellular populations and diverse cell phenotypes. Each cell and their organelles are modeled with different shapes and distinct distributions of biomarkers over each shape, which can be affected by the cell's microenvironment, demonstrating the importance of good cell placement (e.g. in clusters, overlapping existing cells) [26].

The 'CellOrganizer' toolbox was developed based on laboratory data and using machine-learning techniques to generate the entire cell, including structures such as the nucleus, proteins, cell membrane and cytoplasm components [27]. Although the learn-based model was capable of extracting a very precise shape model, it cannot be described in precise mathematical terms [28].

Most image generators have focused on the simulation of morphological features and spatial information of the cell. Morphological information can suffice to create multidimensional images, but it cannot simulate time-lapsed multimodal and functional images, where important time-dependent processes are present. To simulate such images of bacterial cells, the 'miSimBa' (Microscopy Image Simulator of Bacterial Cells) tool has been under development [29]. The simulated images can reproduce spatial and temporal bacterial time-dependent processes by modeling cell growth, division, motility and morphology: shape, size and spatial arrangement [29]. Relevantly, these simulation tools can also be used to generate "null-models" [30], to study statistical patterns in absence of a particular mechanism (e.g. removing the nucleoid to study how it influences the spatial distribution of protein aggregates).

2.2 Cell Tracking

Several tracking methods have been proposed, differing on how they process available object features, type and number of tracked objects [10]. In order to decide which approach to follow, the object's representation, defined during the segmentation process, must be taken into account. Objects can be represented through points, geometric shapes, silhouette and contour, articulated shape model or skeletal model, leading to different developmental approaches [10]. Tracking methodologies were divided into three main categories: Point Tracking, Kernel Tracking and Silhouette Tracking [10].

Objects in Point Tracking are represented by points and tracked based on their position and motion. The main issues of this methodology are the presence of occlusions and the entries and exits of objects in the field of view. This category has been divided in Deterministic and Statistical methods. Deterministic methods associate each object with the application of motion constraints, while statistical methods take into account random perturbations and noise during the tracking process [10]. The Nearest-Neighbor (NN) algorithm is the source of all deterministic approaches and uses only the distances between objects in k and k-1, matching the objects with the smallest distances. This distance can be based on position, shape, color and size [31].

An efficient visual object tracking algorithm was proposed by [32] that combines NN classification with descriptors based on the scale-invariant feature transform, effi-

cient sub-window search and an updating and pruning method to achieve balance between stability and plasticity. This method successfully handles occlusions, clutter, and changes in scale and appearance.

The probabilistic data association filter (PDAF) and the joint probabilistic data association filter (JPDAF) are the basis for the statistical methods. PDAF uses a weighted average of the measurements as input, modeling only one target and considering linear dynamics and measurement models. JPDAF is an extension of PDAF, allowing multiple target tracking. The assumptions are the same when calculating the target's association probabilities jointly. In both methods, if the model is linear, then the Kalman Filter has a relevant influence. One of the problems of these methods is the incapacity to recover from errors, because only the last measurement is used [31]. The Kalman filter is an optimal estimator, which means that it assumes parameters from indirect, inaccurate and uncertain observations and if all noise is Gaussian, the linear Kalman filter minimizes the mean square error of the estimated parameter. This filter is widely used to obtain the optimal state estimate [31].

A different method [33] combining the JDPAF and a particle filtering [34] was proposed and was named 'Monte Carlo JPDAF'. This method uses three models: the first with near constant velocity, the second with near constant acceleration and a third with both models, which achieved the best performance.

Another statistical method is the multiple hypothesis tracking (MHT), which is one of the most used with point features, but has computational limitations both in time and memory [9]. This method postpones data association until enough information is available. The MHT starts by formulating all possible hypotheses, which develop into a set of new hypotheses each time new data arrives, generating a tree of hypothesis [31]. For each hypothesis, the position of the object in the next frame is predicted and then compared with the measurements, calculating their distance. The associations are made for each hypothesis, generating new hypotheses for the next iteration [10]. The tree of hypotheses should be cut, because it grows exponentially with the measured data. This can be done by clustering, i.e., measurements are subdivided into independent clusters. If a measurement cannot be associated with an existent cluster, a new one is created. Another way of cutting the tree is pruning, meaning that as new iterations are added, a part of the tree is deleted [31].

Unlike PDAF and JPDAF, the MHT method can deal with objects entering, exiting and being occluded from the field of view. Kernel Tracking can be done using templates and density-based appearance models or multi-view appearance models. Templates use basic geometric shapes, while multi-view models encode different views of the object. Mean shift and KLT (Kenade-Lucas-Tomasi) are examples of template and density-based appearance models [10].

In mean shift, the appearance of the objects being tracked is defined by histograms. Similarities are measured using the Bhattacharyya coefficient [35] and the Kullback-Leibler divergence [36]. The process tries to increase similarity between histograms, by repeating each iteration until they converge [37].

KLT is an optical-flow method, which uses vectors to show the changes in the image (i.e. translation). A version of this method was proposed in which the translation of a region centered on an interest point is iteratively computed. Then, the tracker evaluates

the tracked patch, computing a transformation in consecutive frames [38]. These methods are effective while tracking single objects, but have problems dealing with multiple objects. Silhouette Tracking consists in using precise information about the shape of the objects, using Shape Matching and searching for an object silhouette and its model in each frame. Each translation from frame to frame is handled separately by finding corresponding silhouettes detected in two consecutive frames. Another approach is based on the evolution of the object contour, connecting the correspondent objects by state space models or by minimizing the contour energy [10].

When tracking objects, one usually obtains multiple measurements. The incorrect ones are referred to as false measurements or clutter. The measurement with highest probability of being originated from the tracked object is then selected. If the algorithm selects the wrong measurement or if the correct measurement is not detected, a poor state is estimated. To solve this issue (reducing the computational cost), a validation region (measurement gate) is selected. The measurement gate is a region in which the next measurement has a higher emergence probability [31].

3 Methodologies

3.1 Implementation of the Image Generator - Tool Interface and Basic Functionalities

The image generator interface and the tracking methods were implemented using the C# language from Visual Studio 2015. This sub-section focuses on the implementation of the image generator and its basic features. In order to facilitate the analysis of the tracking algorithms an intuitive interface was designed. The time-series generator allows the user to change a number of settings such as the number of objects, frames, clusters, and their features. The generator automatically creates a csv file containing the object's properties (position in x and y coordinates and a shape-related factor called "morphology", which is a rational number between 0 and 1 as defined in the Object Shape Sub-Section). The tool interface is shown in Fig. 1. At the top row of the window there are frame handlers, to advance forward and backward in the time-series, or to go directly to a specific frame. The "Time-Lapse" button reproduces the full time-series with a frame-rate of 25 frames/second.

The left bar contains the boxes to write the desired width and height of images, in pixels. The user can also choose the number of objects in each frame, and the total number of frames. The "Maximum Velocity" is the maximum distance, in pixels, that an object can travel between frames, while the "Maximum Morphology Difference" is the maximum difference of the "morphology" factor that an object can have between frames, in percentage. The "Physical Move" button controls the option of giving objects physical limitations to their kinetics. If it is selected, each object has a velocity and orientation assigned to it, meaning that its position dynamics will depend on these two variables. If it is not selected, objects will move arbitrarily between frames.

One can also select "Allow Entries/Exits", to allow the objects to enter and exit the image limits. If unselected, objects collide and are reflected by the edges of the image when reaching them. When the option "Allow Occlusions" is selected, objects move

without restrictions due to superposition between them. If it is not selected, objects collide between them similarly as when colliding with the edges.

Objects clustering can also be forced checking the "Create Clusters" option. When selected, all objects of each cluster have the same physical features. In this setting, "Physical Move" is automatically selected and "Allow Occlusions" is deselected, blocking the correspondent checkboxes. The button "Cluster Properties" (shown in Fig. 5) leads to a new window with the options for clusters' creation. Here, the desired number of clusters, objects per cluster, and size of the clusters in pixels can be selected. It is also possible to choose between two types of objects' kinetics: "Follow the Leader" and "Alternative Movement". The application of "Cluster Centre Force" and its strength are shown in Fig. 6 and explained in the Sub-Section Cluster Creation.

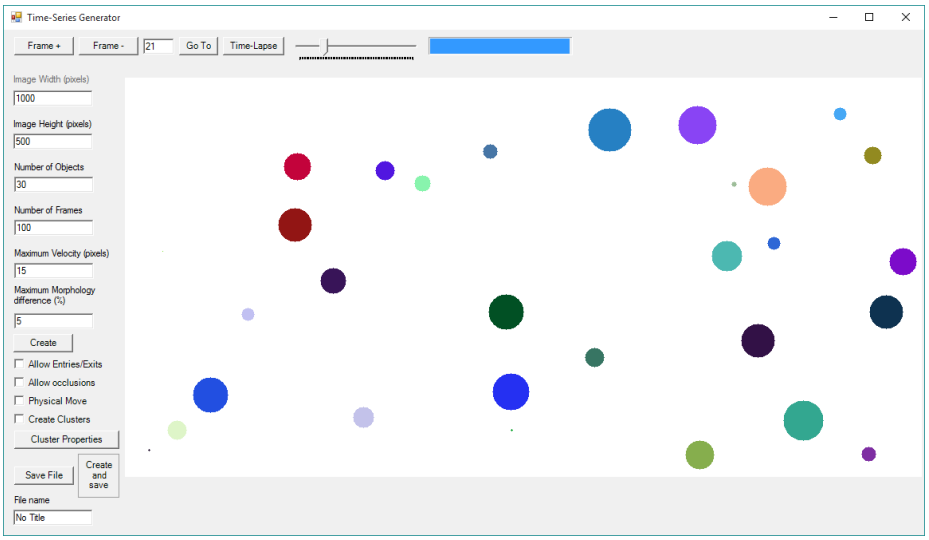

Fig. 1. Image Generator Tool interface.

3.2 Object Modeling

This sub-section focuses on the modeled features, namely object shape, movement, growth, division and clustering, which were improved from the previous toolbox towards a realistic simulation of the bacterial cell spatial and temporal organization.

Object Shape.

To create a realistic simulation of bacterial cells, we first need to investigate how they are classified by their shape. Bacterial cells can have a spherical shape (coccus) a rod-shape (bacillus), while other bacteria have shown a vast diversity of shapes, such intermediate shapes (coccobacillus) or curved/corkscrew shapes (spirochete, spirillum and vibrio), or even square and star shapes, each of them with its specific purpose [39, 40].

Bacteria can also have a wide range of cell sizes (volumes that range from 0.02 to 400 µm3), where even a vast variability can be observed within the same species [41, 42]. These variations can be explained due to cell adaptation to external factors, such as lack of nutrients leading to starvation, situations of extreme temperatures (low and high) or of extreme dryness [42].

A typical bacterial cell envelope is mainly composed by a cytoplasmic membrane and peptidoglycan (also known as murein) cell wall. Bacteria can also be divided in two groups regarding a fundamental difference in the cell envelope: Gram-negative and Gram-positive bacteria. In the first group (which is the case of *E. coli*) a bacterial outer membrane is also present (with intercalating pore-forming proteins, called porins), with lipopolysaccharides connected to the exterior of that outer wall. The interior of the outer wall is then connected to a very thin murein wall by a lipoprotein [43]. In the second group (which is the case of human pathogenic bacterium Streptococcus pneumonia), the cell envelope consists of a very thick murein wall (sometimes more than 10 times thicker than the first group) with teichoic acids spread across the murein. The shape is maintained and determined by the way murein is incorporated during cellular elongation, especially in rod-shape organisms, such as *E. coli* [44] and *B. subtilis* [45], as the murein is the main cell wall structure that supports the stress from the outside [46], as computational physical models have been develop to study how defects in the murein can affect *E. coli* shape (and the shape robustness to murein damage) and how different murein defect patterns can build bacterial shape patterns such as curved rods and spirochaetes [47]. Along with the cell wall, other cytoskeleton proteins are associated with bacterial shape, such as FtsZ (tubulin homologue), MreB (actin homologue) and crescentin [39, 45].

Fig. 2. Example of bacterial cell shapes. Spherical shape (coccus) in dark gray, a rod-shape (bacillus) in orange, intermediate shape (coccobacillus) in green and curved shapes (spirochete, spirillum and vibrio) in blue.

In the first version of this generator [48], objects were just represented by circles and the morphology factor (radius), which only represented coccus shaped bacteria. There was a conversion factor that determines the maximum radius of the objects (corresponding to morphology value 1). By default, this factor was initialized at 30. The development towards realistic bacterial cells will involve the representation of objects with this variables: *Object_ID* (identifies each object), *MajorAxisSize* (size of Major Axis), *Orientation* (defines the orientation of the Major axis), MinorAxisSize (size of Minor Axis), *Curvature* (0 if you want to create line objects and 1 if both ends of the Major

Axis touch, transforming the long bacteria into a circle) *IDPixelList* (this variable populates all pixels that correspond to the object), *Centre* (center of mass of the object), *Division* (this value starts at 0, changes to the time step where the division event occurred), *Parent* (equal to its *Object_ID* in every time-step except in the time-step after a division event, which is equal to the parents *Object_ID*). With these new parameters we are able to change the shape of the cell towards more realistically bacterial shapes. Cells with similar *MajorAxisSize* as *MinorAxisSize* will have coccus shape, when we increase the *MajorAxisSize*, we will get intermediate shapes (coccobacillus) and with large increases of *MajorAxisSize* we will have bacillus shapes. Using the *Curvature* and large *MajorAxisSize* we will create curved shaped objects. These properties are populated for each time-step of the simulation and can be changed by events such as cell growth, division, motility and clustering, as explained in the next sub-sections.

Object Growth.

Bacterial cell cycle is normally divided in three stages, specifically a period between its "birth" and the initiation of DNA replication, a replication period when the cell increases its mass and size (Cell Growth) and, finally, a binary fission process into two new daughter cells (Cell Division), which is repeated over the next generations [49].

The creation of new murein polymer can lead to cell growth through cell elongation, as murein is inserted in the sidewalls at the middle of the cell or at the poles. The creation of the division septum at the mid-cell then leads to a division event (this is also the main process for cell growth in spherical cells, where cell elongation does not occur), where two daughter cells are created [39]. Each of those processes have their own protein and enzymatic apparatus, working in specific places of the cell wall [39, 45]. The FtsZ cytoskeleton protein along with various other proteins create the division septum at the middle of the cell (as two proteins MinC and SlmA that are present in the rest of the cell, inhibit the assembly of the FtsZ ring required for division [50].

In the first version of this generator [48], the morphology shape-related factor called was set at 0.05 (this value was chosen to emulate biologically inspired objects that slowly change their shape over time). Although this process emulates how other cell shapes (bacillus, coccobacillus, vibrio) change their cell size, this actually needs to be changed in truly spherical shaped cells (cocci) as they do not have an elongation process [51], but create a division septum at mid-cell, which allows them to create two daughter cells roughly of the same size of the parent cell due to entropic forces [39]. For the remaining shapes, we implement the creation of new pixels along the Major Axis as the growth process.

Object Division.

In the first version of this generator [48], no division process was implemented. This new version has implemented object division This feature is intended to be an approximation to living cell proliferation, where a parent cell "splits" in half, originating two daughter cells. In this specific case, since objects are represented only by circles, not

by complex shapes, division consists in splitting an object with a morphology factor m into two objects with a factor m/2.

There was a factor named “Division Probability”, measured in percentile that defines the probability of occurring a division for each object, in each frame of the time-series, which happens stochastically. The daughter objects inherit from the parent the physical parameters share the same cluster force (if inside a cluster). An example of an object division is shown in Fig. 3.

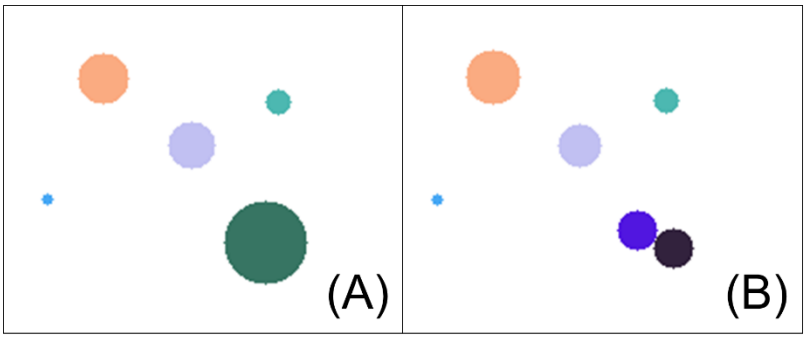

Fig. 3. Example of object division from frame (A) to frame (B)

Object Motility.

Bacterial growth as a colony can also be dependent on the capability to move in the direction of more favorable conditions, which at its basic form is normally associated with Brownian random movement or active movement towards a specific gradient, e.g. chemicals (chemotaxis) and temperature (thermotaxis) [52].

According to the user’s selection, objects can have movement respecting a number of physical rules. If this option is deactivated, objects will move arbitrarily through the image. In each frame, each object can move to a new x and y coordinates by an arbitrary distance that cannot be higher than the “Maximum Velocity” value in pixels.

If entries and exits are deactivated and if an object is heading to the image boundary, it is reflected respecting Snell’s Law, as seen in Fig. 3 causing a change in the angle’s direction of movement. If occlusions are deactivated, when two objects are about to collide, they change to opposite orientations in an approximation to the reflection laws, but ignoring differences in their morphologies.

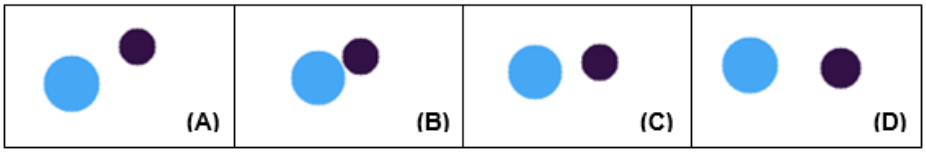

Fig. 4. Collision between objects with "Physical Move". Objects in: (A) Frame 10; (B) Frame 16; (C) Frame 19; (D) Frame 23.

With occlusions and “exits and entries” also deactivated, objects will avoid the positions where they collide with other objects or go out the image boundaries, searching for a position considering these limitations and the maximum distance they can move between frames. If the user chooses to give objects “Physical Move”, in addition to previous features, each object will have a velocity and an orientation assigned to it, meaning that their position dynamics will depend on these two values. In each frame, each object will have new x and y coordinates distanced “d” (no bigger than “Maximum Velocity”) from the previous frame coordinates, direction “o” (between 0 and 2pi radians), with both components using an independent random variable, consistent with the Brownian random movement. Collisions between objects might need to be reconsidered as bacterial cells tend to create clusters when they bump with other cells, and not move away from those cells.

Cluster Creation.

In terms of spatial arrangement, bacteria can be organized in single forms or be grouped in pairs (diplo prefix), in chains (strepto prefix). Cocci bacteria can also organize in groups of 4 (tetrad), 8, 16 or 32 (sarcinae) or in grape-like clusters (staphylo prefix). Bacilli bacteria can organize in palisade structures (side by side) or can be in unstructured spatial clusters [40].

When selecting the option “Create Clusters”, the Generator will create a time-series with the number of clusters, objects and size of cluster chosen by the user. These options (shown in Fig. 5) must be consistent and take into consideration the image size.

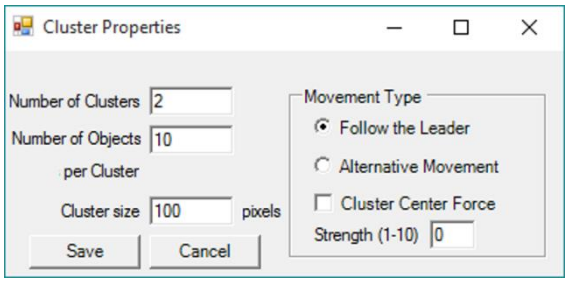

Fig. 5. Interface options for cluster properties.

In “Alternative Movement” (as shown in Fig. 6-A) all objects of each cluster have the same physical parameters, which means that they move in the same direction with the same speed (with a small independent arbitrary component).

In the “Follow the Leader” movement mode (as shown in Fig. 6-B), each cluster has a leading object. The characteristics of the other objects of the same cluster are dependent on the leader’s behavior. The leader “receives” the physical parameters at first frame (velocity and orientation) and at each frame the other objects of its cluster will move in the leader’s direction, minimizing the distance to it, but respecting the “non-collision” rule. If two objects from different clusters collide, one of them will start belonging to the other cluster. This may cause the “merging” of clusters.

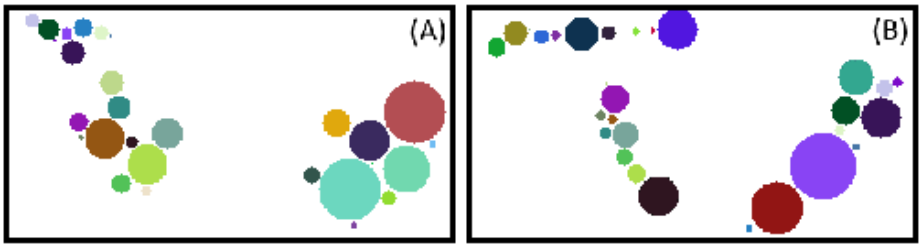

Fig. 6. Exemplificative frames of (a) 'Alternative' Movement (b) 'Follow the leader' Movement.

The "Cluster Centre Force" feature is exclusively for "Alternative Movement" that creates an attraction force at the cluster's center, with a selectable strength selected by the user. This force keeps cluster's objects together, even when colliding with the image borders or other objects. Increasing the strength, the objects will move faster to the cluster's center. In this mode of motility, when objects from different clusters collide, they will be "left behind" by their cluster until they can join it again.

3.3 Tested Tracking Algorithms

In this Section, we give a small introduction to Nearest-Neighbor Algorithms that were used to test our image generation tool, namely the Simple Nearest-Neighbor (NN) and the Nearest-Neighbor with Morphology (NNm) Algorithms. We also introduce the Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which is mainly used to track clustered objects.

Simple Nearest-Neighbor Algorithm.

The first tracking algorithm tested was the Simple NN. This method only takes into consideration the position of each object in each frame of the time-series, and uses the Euclidian Distance between points to find matching objects between frame n and n+1. Being d_p the distance between two objects:

$$d_p = \sqrt{(x_n - x_{n+1})^2 + (y_n - y_{n+1})^2} \quad (1)$$

Where x_n and y_n are the positions of each object in frame n and x_{n+1} and y_{n+1} are the positions in frame n+1. Having the distance between each object in frame n and all objects in frame n+1, correspondences are made based on the minimum distance. The object in frame n+1 closer to each object in frame n is assigned to it. If two objects in n+1 are assigned to the same object in n, the closer object is assigned, until all correspondences between frames are unique [31].

Nearest-Neighbor with Morphology algorithm.

The NNm algorithm accounts not only for the differences between the positions of each object in each frame, but also for a shape-related factor, called morphology. This algorithm calculates the distance percolated by each object between frames n and n+1 using

equation (1). Being m_n the morphology of each object in frame *n*, and m_{n+1} the shape factor in *n*+1, the difference, d_m, between these variables is calculated by:

$$d_m = |m_n - m_{n+1}| \quad (2)$$

The total difference, d_t, between each object in each frame pair is given by (3) with α and β being the weights given to each partial distance. Here different weights are used (as presented in the Results section), in order to study the best way to combine them:

$$d_t = \alpha \cdot d_p + \beta \cdot d_m \quad (3)$$

Cluster Tracking.

Identifying clusters is one of the most complex issues of image characterization [53]. In this work, the problem lays in tracking objects knowing that they are grouped in clusters. Since bacteria often group this way, the goal is to find a method that improves tracking of clustered objects. One of the main problems of clustered objects is illustrated in Fig. 7-A. Using NN (or NNm) to track these frames, the algorithm will immediately misidentify at least two of the objects of frame *n*+1. This will occur in objects 1' and 3', and it happens because their position in *n*+1 is exactly the same that objects 2 and 4 have in *n*.

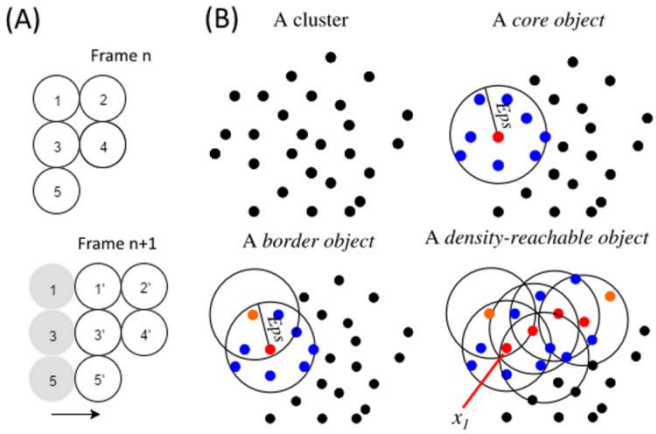

Fig. 7. (A) Example of a possible misidentification using the NN Algorithms. (B) 'MinPts' is defined as the minimal number of neighborhood objects, and Eps as the neighborhood radius, a core object (Red) is defined when its local density is higher than 'MinPts' and a border object (Orange) in its local density is less than 'MinPts'. Two density-reachable objects are defined if a chain of core objects exists with distances between them smaller than Eps. Adapted from [54].

To solve this problem we implemented a tracking algorithm that considers the cluster's features and its singularities. The first step of this method to track clustered objects is to correctly identify the clusters in the image and the objects belonging to each of them. The adopted method was the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [55] in its revised version [54]. This method formalizes the notion

of "cluster" and "noise", using the definition of density to characterize clusters, meaning that to define a cluster, the density of the neighborhood of each point has to be higher than a given threshold. '*MinPts*' is the minimal number of objects in the neighborhood, and Eps is the neighborhood radius (see Fig. 7-B).

Objects can be divided into three categories: core, border and noise (see Fig. 7-B). An object is a core object if its local density is higher than '*MinPts*'. It is considered a border object if its local density is less than '*MinPts*' and it belongs to the neighborhood of a core object. An object is classified as noise if in its Eps radius there are less than '*MinPts*' objects and none is a core. Finally, we identify two density-reachable objects if there exists a chain of core objects between them (see Fig. 7-B), with distances between them smaller than '*MinPts*' [54].

This approach improves clustering identification when the data has dense adjacent clusters [54]. They also introduced the concept of core-density-reachable objects, which is similar to the chain of density-reachable objects, but cutting border objects from chain's ends and staying unclassified until all core objects are identified [54].

The algorithm has two main steps: '*dbscan*' and '*ExpandCluster*'. The first step lies in covering each object and running '*ExpandCluster*' if the object is unclassified. Then, it returns all objects that are core-density-reachable from that one. If it is a core object, a cluster is produced. If it is a border object, it has no core-density-reachable objects, and proceeds to the next one. After all chains from the core object are known, it is assigned to its best density-reachable chain and all border objects.

After identifying the clusters in all frames with DBSCAN, a novel algorithm for object tracking was developed. This algorithm assumes that objects are grouped and move in clusters, treating each cluster as a separate individual while tracking. The first step (with all clusters identified) is to isolate the clusters and calculate their centroid, in coordinates x and y:

$$x_{centroid} = {\sum_{i=1}^{N} x_i}/{N} \tag{4}$$

After all centroids are calculated, they are processed as objects, since they have their own coordinates. The NN algorithm is then applied to these coordinates, tracking each cluster individually and resulting in a sequence of results similar to object tracking.

4 Results and Discussion

We generated several time-series that can be used as a benchmark to test tracking algorithms. For this, we simulated examples with different starting number of objects (20 to 160) and 'Maximum Velocity' (V=5, 10, 15, 20 and 30).

The generated images have a 1000x500 pixel size (first and second experiment) and 1500x100 (third experiment). The implemented Tracking Algorithms automatically processes the csv files with the objects' true positions produced by the Image Generator. The detected object tracking is then compared with the gold standard. In this comparison, a False Positive (FP) is counted when one object is incorrectly tracked from one frame to another and a True Positive (TP) is accounted when one object is tracked correctly between two consecutive frames. It is important to notice that errors that occur

in the beginning of the time series are typically propagated through the entire sequence. We present in the following tables the tracking error (false discovery rate), calculated as FP/(FP+TP).

4.1 Simple Nearest-Neighbor algorithm

We tested 10 time-series of 100 frames for each example with different objects and different maximum velocity. In Table 1, we present the tracking performance of the Simple NN algorithm, based on the ground-truth produced by the image generator. The tracking error is calculated on every frame and accumulated until the end of the time-series. In this case, the morphology shape-related factor called was set to 0.05 (this value was chosen to emulate biologically inspired objects that slowly change their shape over time). The results from Table 1 show that this simple algorithm can handle the increase in the number of objects while keeping a small velocity, and that when raising the velocity to 20 and 30 the tracking performance was significantly reduced.

Table 1. Tracking errors of the Simple Nearest-Neighbor Algorithm.

Obj.	V=5	V=10	V=15	V=20	V=30
20	0,00	0,92	1,06	4,19	19,20
40	0,26	1,27	3,23	5,93	24,01
60	0,06	1,58	5,63	12,38	39,66
80	0,24	1,84	6,62	15,74	45,06
100	0,27	1,20	7,85	19,94	49,76
120	0,22	1,69	10,57	21,16	51,86
140	0,55	3,71	14,16	26,57	58,07
160	0,42	4,12	14,91	33,74	63,89

4.2 Nearest-Neighbor with Morphology Algorithm

In this second experiment, we show how tracking taking into account the morphology of the object can be helpful in the worst case scenario of the last experiment. In Table 2, we present the results of the tracking performance of the NNm Algorithm. In this case we also produced 10 time-series of 100 frames for each example with different objects and different maximum velocity, but also with distinct morphology factors.

We tested the algorithm in two configurations; the first giving a 60% importance to the calculated distance between objects (α factor in equation 3) and 40% to the calculated morphology difference (β factor). For the second configuration we used 40% for α and 60% for β. The impact of the shape-related factor was also studied using both 0.05 and 0.1. For this section we extended our results [48] to include lower velocities and less objects when comparing our analysis against the simple NN algorithm. From Table 2, we observe that tracking results are improved by using the NNm Algorithm (e.g. in the worst case scenario the error percentage was reduced from 64% to 47%) for the m factor =0.05 case, but at lower velocities, the Simple NN algorithm achieves similar results (compared with NNm) even with a large number of objects.

Table 2. Tracking errors of the Nearest-Neighbor with Morphology Algorithm.

	α = 60% and β=40%									
	m factor= 0.05					m factor= 0.1				
Obj.	V=5	V=10	V=15	V=20	V=30	V=5	V=10	V=15	V=20	V=30
20	0.00	0.92	0.00	2.27	11.28	0.00	0.00	1.43	0.37	14.17
40	0.00	0.46	2.45	3.36	18.10	0.00	0.33	2.00	8.71	21.06
60	0.06	1.08	3.20	8.82	30.61	0.34	0.31	4.10	8.27	27.12
80	0.24	1.43	4.66	11.00	34.27	0.00	0.88	5.40	13.76	34.87
100	0.27	1.47	6,02	14.71	41,60	0.20	1.96	6,03	17,79	40,55
120	0.00	1.10	6,27	14,92	42,05	0.13	1.74	9,24	19,68	44,45
140	0.22	2.19	9,29	18,34	48,96	0.27	2.66	8,34	21,59	48,90
160	0.13	3.32	10,32	25,49	55,35	0.19	3.03	10,26	25,39	55,32
Obj.	α = 40% and β=60%									
20	0.00	0.66	0.00	2.63	6.99	0.00	0.00	1.43	0.02	8.90
40	0.00	0.46	1.50	3.16	14.92	0.00	0.48	0.45	5.52	15.66
60	0.06	0.84	1.62	6.66	24.02	0.34	0.02	2.41	7.13	22.69
80	0.24	1.37	3.10	7.30	26.65	0.00	0.65	4.00	9.90	27.80
100	0.19	0.84	4,26	10,57	33,37	0.20	1.07	3,96	12,07	32,44
120	0.00	0.84	4,53	10,80	33,95	0.13	0.79	6,39	14,07	37,09
140	0.18	0.89	6,36	14,58	39,30	0.25	1.61	5,34	15,18	41,36
160	0.13	1.88	7,27	20,78	46,81	0.19	2.03	8,06	18,93	49,38

It is important to note that, as most of bacterial cells in live-cells imaging are placed in agarose gel, where they do not move very fast, but they are able to grow and create large clusters of cells, in cells that have large movement capabilities, other tracking algorithms need to be used and compared. We also should note that the second configuration (40% for α and 60% for β) gave better results than the first one, so giving more importance to the morphology factor, improved the results (comparing the results for the same number of objects and same velocities). Results might still be improved by using different configurations of the α β parameters, so this is will be one of our future efforts in the improvement of tracking algorithms.

4.3 Cluster Tracking

The Create Clusters property was used to test the same tracking algorithms (Simple NN and NN with Morphology Algorithms with α=40%). The simulated parameters were: number of clusters (1, 5 and 10), number of objects per cluster (10 and 15), maximum velocity (5 and 10), Alternative Movement, Center Force (4) and morphology factor (0 and 0.05). The tracking results are presented in table 3. For the Cluster creation, we used 10 time-series (and averaged the results) of 200 frames and calculated the object tracking error on every frame accumulated throughout the time-series.

The DBSCAN algorithm tries to separate each cluster in every frame. Therefore, if the number of clusters is the same between the actual frame and the previous one (t and t-1), then they are matched using NN, treating them as isolated objects and aligned using their centroids. If the number of clusters changes, the first step is skipped and the number of objects inside each cluster is checked. When, inside a cluster, there are more objects in t then in t-1, these 'extra' objects are labeled as 'Possible Entry'. If there are

fewer objects, they are labeled ‘Possible Exit’. This tagging is temporary and compares the "Possible Exit" features to the features of all other objects of the frame t-1, linking it to a "Possible entry" in another cluster (meaning that it left one cluster to join another), classifying it as noise, or as an object leaving the image. The main difference between DBSCAN 1 and DBSCAN 2 algorithms is that, in the first, this classification is done after the tracking and in the second it is done before the tracking, equalizing the number of objects between the clusters.

Table 3. Tracking errors, within clusters with different properties, using the Simple and Morphology NN Algorithms with different number of clusters (1 to 10), different number of objects per cluster (5 to 15), and different maximum velocities (2 to 10).

Simple NN Algorithm							
Nº of Clusters	Obj. / Clusters	m factor= 0			m factor= 0.05		
		V=2	V=5	V=10	V=2	V=5	V=10
1	5	2.95	0.58	2.62	1.93	2.32	14.23
	10	3.32	7.79	30.42	0.93	9.88	23.33
	15	4.63	11.74	50.91	2.94	10.74	38.06
3	5	0.05	2.40	7.69	0.00	4.57	9.01
	10	1.07	7.11	27.83	2.20	9.07	30.08
	15	3.05	14.74	43.77	2.76	16.77	45.44
5	5	0.23	2.22	6.25	0.72	3.28	9.74
	10	0.70	7.48	34.71	1.57	10.95	31.89
	15	3.06	17.43	45.22	3.53	16.06	44.51
7	5	0.58	2.55	12.78	1.04	1.95	14.52
	10	1.58	11.21	33.76	1.81	11.78	40.35
	15	3.14	19.81	48.55	4.01	17.75	48.96
10	5	0.99	3.39	17.81	0.25	5.40	17.13
	10	1.95	12.20	38.26	1.52	11.64	42.47
	15	3.84	21.14	53.90	4.87	23.52	57.34
NN with Morphology (α = 40% and β=60%)							
1	5	0.00	0.00	0.00	1.93	0.00	7.92
	10	0.01	1.27	4.88	3.04	5.52	13.83
	15	0.18	3.76	21.14	1.75	4.63	20.76
3	5	0.38	1.08	1.66	0.00	1.23	4.08
	10	1.18	1.26	10.33	0.03	2.13	12.52
	15	1.81	5.29	20.24	1.92	8.12	22.44
5	5	0.00	1.78	2.58	0.10	0.68	5.77
	10	0.71	1.80	12.98	0.15	4.69	15.93
	15	1.54	7.16	20.77	0.93	5.95	22.07
7	5	0.20	0.41	2.82	0.41	0.35	5.13
	10	0.78	3.92	15.08	0.48	3.60	17.84
	15	1.22	8.14	25.78	1.99	6.86	27.11
10	5	0.04	0.97	6.93	0.15	2.31	7.20
	10	0.48	3.78	16.15	0.54	4.55	19.71
	15	1.11	8.73	28.36	2.22	10.13	34.12

Results from both DBSCAN Algorithms are presented in Table 4 (m factor =0.00) and Table 5 (m factor =0.05).

Table 4. DBSCAN1 (DB1) and DBSCAN1 (DB2) tracking errors comparison for different number of clusters, objects per cluster, and maximum velocities, with m factor =0.

mmd = 0.00							
		Vmax=2		Vmax=5		Vmax= 10	
Clusters	Objects/ Cluster	DB1	DB2	DB1	DB2	DB1	DB2
1	5	0.00	0.00	0.00	0.00	0.16	0.16
	10	0.00	0.00	5.55	4.67	2.97	2.97
	15	0.24	0.24	1.92	2.59	12.94	12.96
3	5	3.10	2.47	5.76	6.01	8.81	8.72
	10	0.70	1.02	4.86	4.89	11.28	10.70
	15	0.83	0.83	3.86	3.86	16.44	16.34
5	5	5.01	5.20	2.46	2.58	15.87	16.80
	10	1.44	1.04	5.43	5.54	13.71	14.56
	15	0.27	0.27	5.84	5.74	19.29	19.35
7	5	2.05	1.89	4.82	5.29	6.37	6.49
	10	2.47	2.23	4.52	4.81	16.18	16.51
	15	0.83	0.83	8.44	8.60	25.45	25.36
10	5	3.15	2.82	9.50	9.03	11.90	12.11
	10	2.45	3.33	5.81	6.00	17.55	17.57
	15	1.24	1.24	8.65	8.65	28.29	28.29

Table 5. DBSCAN1 (DB1) and DBSCAN1 (DB2) tracking errors comparison for different number of clusters, objects per cluster, and maximum velocities, with m factor =0.05.

mmd = 0.05							
		Vmax=2		Vmax=5		Vmax= 10	
Clusters	Objects/ Cluster	DB1	DB2	DB1	DB2	DB1	DB2
1	5	0.96	0.96	1.67	1.67	9.75	9.75
	10	0.00	0.00	9.64	9.64	9.14	7.82
	15	0.85	0.85	3.87	3.87	10.49	10.49
3	5	0.00	0.00	5.06	5.02	13.97	14.89
	10	5.45	5.43	3.06	2.81	9.91	9.83
	15	1.99	1.99	3.78	3.77	19.55	19.63
5	5	2.18	2.69	10.72	11.23	20.79	22.28
	10	1.94	2.33	6.42	7.55	16.61	17.37
	15	0.79	0.79	6.49	6.19	20.84	20.82
7	5	4.07	4.20	7.54	7.43	13.78	12.98
	10	3.37	4.06	4.63	5.46	18.29	18.54
	15	2.00	2.00	7.06	7.22	27.59	27.70
10	5	2.02	1.90	8.33	8.41	13.55	14.38
	10	1.81	2.30	6.42	6.43	21.07	21.42
	15	2.76	2.67	9.91	9.98	34.52	34.52

Comparing Table 3 with Table 1, we can observe that the simple NN cannot handle clusters adequately (for V=10, m factor = 0.05 and 160 objects/cluster, we have a 4,12% error while for V=10, m factor = 0.05, 10 clusters and 15 objects/cluster, for a total of

150 objects we have a 57.34% error rate). From Table 3, we can observe that the NNm algorithm handles much better the cluster creation, giving almost one half of the errors (worst case scenario of 34.12% versus 57.34% for the same configuration).

From Table 4 and Table 5 we can conclude that DBSCAN Algorithms do not improve significantly over the NNm algorithm (Table 3) for the same configuration. A strange behavior for lower velocities was identified in both DBSCAN algorithms, where increasing the objects actually decreased the tracking errors. This behavior is explainable by the higher movement restriction of objects belonging to clusters with larger number objects, but further studies are required to further analyze this behavior. This behavior has not been identified in both simple NN and NNm algorithms.

5 Conclusions and future work

To support high-throughput experiments of single cell imaging, reliable automated image processing methods are required. Although most studies focus on automatic segmentation of cells or cellular structures, in time-series proper object tracking is also necessary, especially because tracking errors propagate, meaning that even small tracking errors (particularly on the initial frames) lead to a high percentage of misidentified tracks overall.

To validate Tracking Algorithms, it is necessary to use a labelled 'ground truth'. Sometimes this ground-truth can be manually obtained, but this strategy is not feasible on a Big Data scenario. A more viable alternative is to generate artificial images by simulating biological cell models. To produce such artificial images, we developed an open source platform that can simulate biologically inspired bacterial systems, by creating cells that of different shapes and sizes, cells that can grow and divide and, cells that can move as a single objects or as clustered objects.

Using this Platform, we evaluated three tracking algorithms (Simple NN, NNm and two variations of the DBSCAN Algorithm). The obtained results show that, for cases with lower maximum velocity, the Simple NN Algorithm was able to track objects even with a significant increase in the number of objects.

Meanwhile, the NNm algorithm can help reducing tracking errors when the velocity is increased. In the example where we forced the creation of clusters, the Simple NN algorithm was unable to handle the increase of number of clusters and objects in a cluster (even for a constant number of objects). On the other hand, the NNm and the DBSCAN algorithms showed similar, significant capabilities to handle large clusters. In the near future, we plan to study and compare other tracking methodologies in different cluster configurations using the proposed framework. Here, the newly developed object division module will be of use to test division tracking in dense clusters.

We expect this open-sourced tool[1] to help future endeavors in the development of new tracking algorithms, as it can produce huge amounts of benchmarked images.

The next steps of our work will be to introduce a new module that generates secondary bodies inside the primary objects, simulating internal cell organelles and structures. A future application will also be made available as a web-based system to improve usability and compatibility.

[1] Tool available at: http://griduni.uninova.pt/Clustergen/ ClusterGen_v1.0.zip

Acknowledgments

Work supported by the Portuguese Foundation for Science and Technology (FCT/MCTES) through a PhD Scholarship, ref. SFRH/BD/88987/2012 to LM, SADAC project (ref. PTDC/BBB-MET/1084/2012) and by FCT Strategic Program UID/EEA/00066/203 of UNINOVA, CTS. This work is also funded by the Academy of Finland [refs. 295027 and 305342 to ASR] and the Jane and Aatos Erkko Foundation [ref. 5-3416-12 to ASR].

References

1. Danuser, G.: Computer vision in cell biology. Cell. 147, 973–8 (2011).
2. Sung, M.-H., McNally, J.G.: Live cell imaging and systems biology. Wiley Interdiscip. Rev. Syst. Biol. Med. 3, 167–82 (2011).
3. Coutu, D.L., Schroeder, T.: Probing cellular processes by long-term live imaging--historic problems and current solutions. J. Cell Sci. 126, 3805–15 (2013).
4. Bonnet, N.: Some trends in microscope image processing. Micron. 35, 635–653 (2004).
5. Frigault, M., Lacoste, J., Swift, J., Brown, C.: Live-cell microscopy - tips and tools. J. Cell Sci. 122, 753–767 (2009).
6. Deshmukh, M., Bhosle, U.: A survey of image registration. Int. J. Image Process. 5, 245–269 (2011).
7. Wyawahare, M., Patil, P., Abhyankar, H.: Image Registration Techniques : An overview. Int. J. Signal Process. Image Process. Pattern Recognit. 2, 11–28 (2009).
8. Meijering, E.: Cell Segmentation: 50 Years Down the Road. IEEE Signal Process. Mag. 29, 140–145 (2012).
9. Tissainayagam, P., Suter, D.: Object tracking in image sequences using point features. Pattern Recognit. 38, 105–113 (2005).
10. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. 38, 1–45 (2006).
11. Selinummi, J., Seppälä, J., Yli-Harja, O., Puhakka, J.: Software for quantification of labeled bacteria from digital microscope images by automated image analysis. Biotechniques. 39, 859–863 (2005).
12. Wang, Q., Niemi, J., Tan, C.-M., You, L., West, M.: Image segmentation and dynamic lineage analysis in single-cell fluorescence microscopy. Cytom. A. 77, 101–110 (2010).
13. Sliusarenko, O., Heinritz, J.: High-throughput, subpixel precision analysis of bacterial morphogenesis and intracellular spatio-temporal dynamics. Mol. Microbiol. 80, 612–627 (2011).
14. Young, J., Locke, J.C.W., Altinok, A., Rosenfeld, N., Bacarian, T., Swain, P.S., Mjolsness, E., Elowitz, M.B.: Measuring single-cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy. Nat. Protoc. 7, 80–8 (2012).
15. Häkkinen, A., Muthukrishnan, A.-B., Mora, A., Fonseca, J.M., Ribeiro, A.S.: CellAging: a tool to study segregation and partitioning in division in cell lineages of Escherichia coli. Bioinformatics. 29, 1708–1709 (2013).
16. Coelho, L.P., Shariff, A., Murphy, R.F.: Nuclear Segmentation In Microscope Cell Images A Hand-Segmented Dataset And Comparison Of Algorithms. In: Proc IEEE Int Symp Biomed Imaging. pp. 518–521 (2009).

17. Xiong, W., Wang, Y., Ong, S.H., Lim, J.H., Jiang, L.: Learning Cell Geometry Models For Cell Image Simulation : An Unbiased Approach. In: Proceedings of 2010 IEEE 17th International Conference on Image Processing. pp. 1897–1900 (2010).
18. Kruse, K.: Bacterial Organization in Space and Time. In: Comprehensive Biophysics. pp. 208–221 (2012).
19. Misteli, T.: Beyond the sequence: cellular organization of genome function. Cell. 128, 787–800 (2007).
20. Svoboda, D., Kozubek, M., Stejskal, S.: Generation of digital phantoms of cell nuclei and simulation of image formation in 3D image cytometry. Cytometry. A. 75, 494–509 (2009).
21. Lehmussola, A., Ruusuvuori, P., Selinummi, J., Huttunen, H., Yli-Harja, O.: Computational framework for simulating fluorescence microscope images with cell populations. IEEE Trans. Med. Imaging. 26, 1010–6 (2007).
22. Ruusuvuori, P., Lehmussola, A., Selinummi, J., Rajala, T., Huttunen, H., Yli-Harja, O.: Benchmark Set Of Synthetic Images For Validating Cell Image Analysis Algorithms. In: Proceedings of the 16th European Signal Processing Conference, EUSIPCO (2008).
23. Lehmussola, A., Ruusuvuori, P., Selinummi, J., Rajala, T., Yli-harja, O.: Synthetic Images of High-Throughput Microscopy for Validation of Image Analysis Methods. Proc. IEEE. 96, 1348 – 1360 (2011).
24. Svoboda, D., Kasik, M., Maska, M., Hubeny, J.: On simulating 3D Fluorescent Microscope Images. In: Computer Analysis of Images and Patterns -12th International Conference, CAIP 2007, Vienna, Austria, August 27-29, 2007. Proceedings. pp. 309–316 (2007).
25. Ulman, V., Oremus, Z., Svoboda, D.: TRAgen: A Tool for Generation of Synthetic Time-Lapse Image Sequences of Living Cells. In: Proceedings of 18th International Conference on Image Analysis and Processing (ICIAP 2015). pp. 623–634. Springer International Publishing (2015).
26. Satwik, R., Benjamin, P., Nicholas, H., Steven, A., Lani, W.: SimuCell : a flexible framework for creating synthetic microscopy images a PhenoRipper : software for rapidly profiling microscopy images. Nat. Methods. 9, 634–636 (2012).
27. Murphy, R.: CellOrganizer: Image-derived Models of Subcellular Organization and Protein Distribution. Methods Cell Biol. 110, 179–93 (2012).
28. Zhao, T., Murphy, R.F.: Automated learning of generative models for subcellular location: building blocks for systems biology. Cytometry. A. 71, 978–90 (2007).
29. Martins, L., Fonseca, J., Ribeiro, A.: "miSimBa" - A simulator of synthetic time-lapsed microscopy images of bacterial cells. In: Proceedings - 2015 IEEE 4th Portuguese Meeting on Bioengineering, ENBENG 2015. pp. 1 – 6 (2015).
30. Gotelli, N.J., McGill, B.J.: Null versus neutral models: what's the difference? Ecography (Cop.). 29, 793–800 (1996).
31. Elfring, J., Janssen, R., van de Molengraft, R.: Data Association and Tracking: A Literature Survey. In: ICT Call 4 RoboEarth Project (2010).
32. Gu, S., Zheng, Y., Tomasi, C.: Efficient visual object tracking with online nearest neighbor classifier. In: Computer Vision – ACCV 2010. Volume 6492 of the series LNCS. pp. 271–282 (2011).
33. Gorji, A., Menhaj, M.B.: Multiple Target Tracking for Mobile Robots Using the JPDAF Algorithm. In: 19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007). pp. 137–145 (2007).
34. Gordon, N., Salmond, D., Smith, A.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. Radar Signal Process. IEE Proc. F. 140, 107–113 (1993).
35. Bhattacharyya, A.: On a Measure of Divergence Between Two Statistical Populations Defined by Probability Distributions. Bull. Calcutta Math. Soc. 35, 99–110 (1943).

36. Joyce, J.: Kullback-Leibler Divergence. In: Lovric, M. (ed.) International Encyclopedia of Statistical Science SE - 327. pp. 720–722. Springer Berlin Heidelberg (2014).
37. Zhou, H., Yuan, Y., Shi, C.: Object tracking using SIFT features and mean shift. Comput. Vis. Image Underst. 113, 345–352 (2009).
38. Shi, J., Tomasi, C.: Good features to track. In: Proceedings CVPR'94., 1994 IEEE Computer Society Conference on. IEEE. pp. 593–600 (1994).
39. Cabeen, M.T., Jacobs-Wagner, C.: Bacterial cell shape. Nat. Rev. Microbiol. 3, 601–10 (2005).
40. Salton, M., Kim, K.: Chapter 2. Structure. In: Baron, S. (ed.) Medical Microbiology. 4th edition. Galveston (TX): University of Texas Medical Branch at Galveston (1996).
41. Zinder, S.H., Dworkin, M.: Chapter 1.7 - Morphological and Physiological Diversity. In: Dworkin, M., Falkow, S., Rosenberg, E., Schleifer, K.-H., and Stackebrandt, E. (eds.) Prokaryotes. pp. 185–220. Springer New York, New York, NY (2006).
42. Koch, A.L.: What size should a bacterium be? A question of scale. Annu. Rev. Microbiol. 50, 317–48 (1996).
43. Höltje, J.-V.: Cell Walls, Bacterial. In: The Desk Encyclopedia of Microbiology. pp. 239–250 (2004).
44. Henning, U., Rehn, K., Hoehn, B.: Cell envelope and shape of Escherichia coli K12. Proc. Natl. Acad. Sci. U. S. A. 70, 2033–6 (1973).
45. Carballido-López, R., Formstone, A.: Shape determination in Bacillus subtilis. Curr. Opin. Microbiol. 10, 611–6 (2007).
46. Höltje, J.-V.: Growth of the Stress-Bearing and Shape-Maintaining Murein Sacculus of Escherichia coli. Microbiol. Mol. Biol. Rev. 62, 181–203 (1998).
47. Huang, K.C., Mukhopadhyay, R., Wen, B., Gitai, Z., Wingreen, N.S.: Cell shape and cell-wall organization in Gram-negative bacteria. Proc. Natl. Acad. Sci. U. S. A. 105, 19282–19287 (2008).
48. Canelas, P., Martins, L., Mora, A., Ribeiro, A.S., Fonseca, J.: An Image Generator Platform to Improve Cell Tracking Algorithms - Simulation of Objects of Various Morphologies, Kinetics and Clustering. In: Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications ISBN 978-989-758-199-1. pp. 44–55 (2016).
49. Wang, J.D., Levin, P.A.: Metabolism, cell growth and the bacterial cell cycle. Nat. Rev. Microbiol. 7, 822–7 (2009).
50. Young, K.D.: Bacterial shape: two-dimensional questions and possibilities. Annu. Rev. Microbiol. 64, 223–40 (2010).
51. Zapun, A., Vernet, T., Pinho, M.: The different shapes of cocci. FEMS Microbiol Rev. 32, 345–60 (2008).
52. Lauffenburger, D.: Effects Of Cell Motility And Chemotaxis On Microbial Population Growth. Biophys. J. 40, 209–219 (1982).
53. Czink, N., Mecklenbräuker, C., Del Galdo, G.: A novel automatic cluster tracking algorithm. IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC. 1–5 (2006).
54. Tran, T.N., Drab, K., Daszykowski, M.: Revised DBSCAN algorithm to cluster data with dense adjacent clusters. Chemom. Intell. Lab. Syst. 120, 92–96 (2013).
55. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: 2nd Int. Conference on Knowledge Discovery and Data Mining. pp. 226–231 (1996).