**Proceedings of the 28th IEEE**
**EMBS Annual International Conference**
**New York City, USA, Aug 30-Sept 3, 2006**



# System for the Validation of Cell-Tracking Algorithms Using On-Demand Simulated Optical Microscope Images

Béchara Moufarrej and Sylvain Martel, *Member, IEEE*

***Abstract*—Computer cell tracking using an optical microscope and an image-processing module has proven to be a valuable technique for the study of cell behavior such as motility and migration. Validation of these techniques is difficult and often done by manual analysis or with the aid of limited and case-specific simulation software. We have developed a general simulation system that returns optical microscope images on demand. The system is designed to work well even in cases of three-dimensional, real-time or very fast moving cells, cases where other validation techniques fail. An evaluation of a popular particle tracker is also presented.**

## I. INTRODUCTION

CELL tracking is a central tool in flow cytometry. Since the developments of optical imagery, optical tracking by image processing and analysis have become commonplace. Typical setups include an optical microscope with an automated stage, a Couple Charged Device (CCD), a programmable control interface and an image analysis module [1-6].

The cell-tracking problem *per se* consists of two subproblems: segmentation and registration [2,5]. Segmentation is the accurate identification and extraction of cells from an image. Registration is the correct association of segmented cells from an image acquired at time $T_n$ with those from an image acquired at time $T_{n-1}$. The tracking problem has been widely addressed in the literature and many tracking algorithms have been developed [5,9]. These are shown to perform more or less well, often with many constraints, but their validation is still problematic. Formal validation has been attempted but has been limited to a particular family of tracking algorithms [6,9]. On one hand, manual validation is prone to error and unrealistic for large amount of data. On the other hand, time-sequence images don't allow testing of real-time strategies and more importantly, are inadequate for testing 3D or single fast-moving cell (e.g., more than 100 times their diameters per second [5]) algorithms. The challenge comes from the fact that the camera frame, or field of view, depends on the tracking algorithm, and hence time-sequence images recorded from the execution of one algorithm cannot be tested on another algorithm. Furthermore, 3D time-lapse images with an optical microscope contain necessarily time discrepancies along the z-axis and prove to be also inadequate, especially for real-time tracking, because of the latency time during acquisition along the z-axis.

For the system presented in this paper, information about cells positions in space and time is known beforehand and is extracted during algorithm execution as an on-demand basis. Tracking results can then be compared with simulation data for exact performance measurements. In the next sections this system will be described and the synthesized images will be shown.

## II. METHOD

Whether in real-time or video-based, optical cell tracking algorithms acquire frame images in a loop as part of their routines. The proposed solution replaces this function call by a request to a *Simulated Images Server* (*SIS*). For simplicity and clarity of reading, a simulated image will be from now on addressed as an *S-image*. *S-images* are constructed in two stages. First the *Position Extractor* (*PE*) calculates the positions of the cells corresponding to the acting *FieldofView* and places cell models at calculated coordinates, and then the *Image Generator* (*IG*) constructs the *S-image* by adding localized and generalized blurring effects. Artifacts can also be added to compensate for optical system or sample aberrations.

The simulator consists hence in two main modules (*PE* and *IG)* interacting with two databases. Secondary modules are used to generate these databases and the complete architecture is shown in Fig. 1. This modular architecture makes the system very portable. Cells can vary much in size and motility and their observation can be done with various visualization techniques. With such a schema in mind, adapting the system to the tracking of a particular type of cell reduces simply to generating the corresponding tracking and cells databases.

This project is supported in part by a Canada Research Chair (CRC) in Micro/Nanosystem Development, Fabrication and Validation, the Canada Foundation for Innovation (CFI), the National Sciences and Engineering Research Council of Canada (NSERC), and the Government of Québec.

B. Moufarrej is with the Nanorobotics Laboratory, Department of Computer Engineering and Institute of Biomedical Engineering, École Polytechnique de Montréal, Campus of the Université de Montréal, Montreal (Quebec) Canada (phone: 514-340-4711 extension 5029; fax: 514-340-5280; e-mail: bechara.moufarrej@ polymtl.ca).

S. Martel is with the Nanorobotics Laboratory, Department of Computer Engineering and Institute of Biomedical Engineering, École Polytechnique de Montréal, Campus of the Université de Montréal, Montreal (Quebec) Canada (Corresponding author: phone: 514-340-4711 extension 5098; fax: 514-340-5280; e-mail: sylvain.martel@ polymtl.ca).

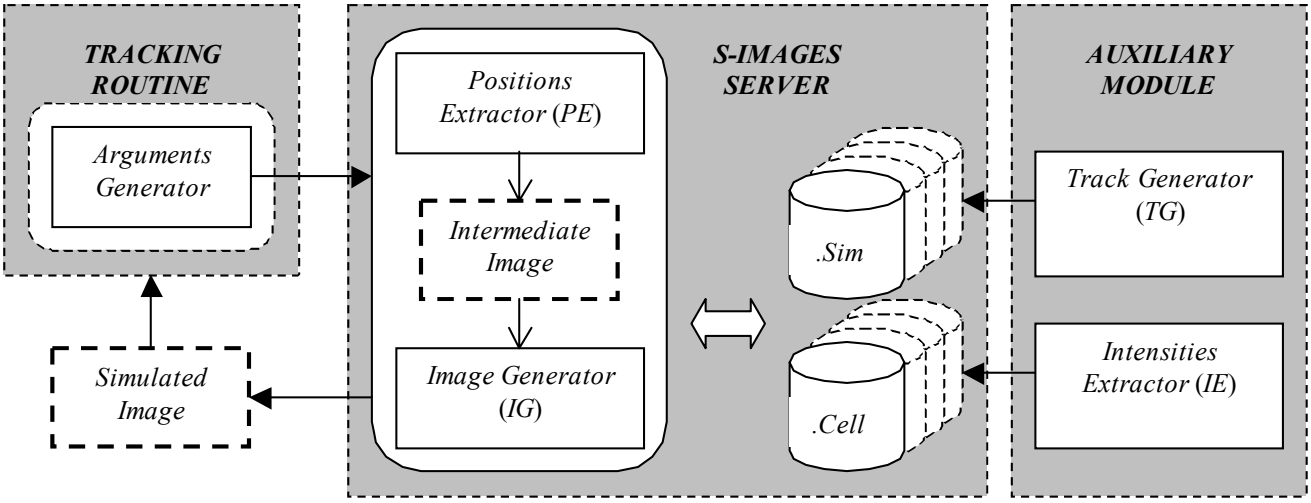


Fig. 1. The architecture of the simulation system. During one cycle, the tracking routine sends a request to the *SIS* with the following arguments: time *T*, stage position and a *simulation*, and receives a simulated image back, at the expense of a minor modification in the image acquisition module.

The different modules are detailed in the following sections, but first some definitions are in order.

### A. Definitions

*Cell*: a *cell* is defined as a set of *[l x w]* gray-scale intensity matrices representing a cell shape and texture (Eq. 1). Such a framework has the advantage of allowing for realistic and unlimited range of geometries inside a bounding box. An auxiliary software module extracts *cells* offline from a representative filtered z-stack image set (i.e., with typical dimensions, illumination and significant noise reduction), and stores them along with other meta-information in *.cell* files that serve as databases for the construction of the *S-images* by the *SIS*.

$$Cell = \{[l \times w]_0 ... [l \times w]_n\}, n \in N \qquad (1)$$

where *l* and *w* are the dimensions of the intensity matrices and n represents the geometric height of the cell in distance units. For *n*=0 the cell reduces to a single matrix, a useful scenario in case of two-dimension tracking algorithms.

*Track*: a *track* is a *cell* mapped to a set of three-dimensional coordinates (Eq. 2), representing a trajectory in the three-dimensional space. *Tracks* are stored in a *.sim* file that serves as a database for the construction of the *S-images* by the *SIS*.

$$Track = Cell \Rightarrow \{(X,Y,Z)_0 ... (X,Y,Z)_n\}, n \in N \qquad (2)$$

where *X, Y* and *Z* are the absolute coordinates of the cells center of mass in the three-dimensional space and *n* represents the duration of the *track* in time units. For *n*=0 the *track* reduces to the cell initial position.

*Simulation*: a simulation is a set of *tracks* (Eq. 3) stored in a *.sim* file. An auxiliary software module generates *.sim* files automatically and is called *Track Generator* (*TG*). A random *TG* has been implemented and is detailed in section D.

$$Simulation = \{track_1 ... track_n\}, n \in N^* \qquad (3)$$

where *n* is the total number of *cells* in the simulated sample, where the simulated sample is a virtual conceptualization of the real cell sample under study.

*Fieldofview*: It is the area of the sample visible through the microscope (Eq. 4). It can be hence defined as an area *A*, function of the microscope's objective lens, associated to the three-dimensional coordinates of the microscope stage's center.

$$FieldofView = A \Rightarrow (X_{stage}, Y_{stage}, Z_{stage}) \qquad (4)$$

where $A_{lens}$ is a linear function of the lens, and $X_{stage}$, $Y_{stage}$, $Z_{stage}$ are the three-dimensional coordinates of the stage's center.

### B. Position Extractor

For the *PE,* a *simulation* acts as an ideal time-sequence series of three-dimensional cell images. It is ideal because only the *tracks* are stored in the *.sim* file. Thus an ideal two-dimensional image is obtained by intersecting one ideal three-dimensional image (i.e., at time *K*) with the *Field of view* (see Fig. 2). That is:

$$I_{PE} = Simulation \cap FieldofView, time = K \qquad (5)$$

that can be shown to reduce to:

$$I_{PE} = \left(Cell_1 \Rightarrow (X,Y,Z)_{K1}\right) \cap \left(A_{lens} \Rightarrow \left(X_{stg}, Y_{stg}, Z_{stg}\right)\right) \cup$$
$$\left(Cell_2 \Rightarrow (X,Y,Z)_{K2}\right) \cap \left(A_{lens} \Rightarrow \left(X_{stg}, Y_{stg}, Z_{stg}\right)\right) \cup \ldots \cup \quad (6)$$
$$\left(Cell_n \Rightarrow (X,Y,Z)_{Kn}\right) \cap \left(A_{lens} \Rightarrow \left(X_{stg}, Y_{stg}, Z_{stg}\right)\right)$$

The algorithm for calculating $I_{PE}$ varies linearly with the total number of *tracks* in the *simulation* and hence its execution is relatively fast even for very large *.sim* files. Hence, the *PE* takes as arguments a *Fieldofview*, a *simulation*, and a time constant *K*, and generates and ideal camera image of two-dimensional *cells* at corresponding positions.

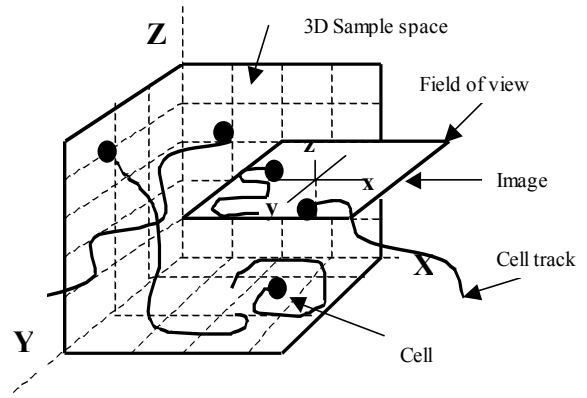


Fig. 2. The intermediate ideal image is the intersection between the camera field of view and the cell geometries at time *T*. (Figure is not to scale.)

### C. Image Generator

The IG module adds a blurring effect to the intermediate image generated by the *PE*. It does so according to the following general blurring model:

$$I_s = I_i \bullet PSF + N \quad (7)$$

where $I_s$ is the blurred *S-image*, $I_i$ is the intermediate ideal image generated by the *PE* module, *PSF* is the point-spread function of the objective and *N* represents the noise introduced during image acquisition.

Similar approaches have been proposed in the literature to compensate for various noise and aberrations introduced by the imaging system. According to [3]:

$$I_s = I_i + PSF + N_g + N_p \quad (8)$$

where $I_s$ and $I_i$ are respectively the simulated and intermediate images, PSF the point spread function describing the optical system and $N_g$ and $N_p$ are Gaussian and Poisson distributions compensating for the detector's noise.

Optional additions in this module include random artifacts additions and regional blurring (e.g., a 0-20% cell-intensities random alteration).

### D. Tracks Generator and Cell Intensities Extractor

These two auxiliary software modules have been implemented in order to automatically generate *.sim* and *.cell* files. A random *TG* generates as part of its *while* loop random variances in *angle* and *speed* and calculates corresponding coordinates for each cycle.

TABLE I
EVALUTATION OF A MATLAB PARTICLE TRACKER

| Optical concentration | Tracking error |
|---|---|
| 10 | 0.0100 |
| 50 | 0.3939 |
| 100 | 0.3088 |
| 250 | 0.6369 |
| 500 | 0.7660 |
| 1000 | 1.3152 |
| 2000 | 1.8544 |
| 5000 | Algorithm Failed |

Optical concentration is in cells/mm$^2$ under the 20x microscope objective. The tracking error is defined in Eq. 9 and is evaluated for a sequence of 10 images. Cells sizes are approximately 5x5 microns$^2$.

## III. RESULTS

In order to test our simulator, we have imagined two scenarios where optical images are acquired with different imaging techniques: Transmitted light Dark Field (DF) microscopy and transmitted light Bright Field (BF) microscopy. The resulting *S-images* are shown in Fig.4. The cells agglomerate in the center part of the screen in BF as a result of simulation parameters. We can observe that DF images present a very high contrast rate. As part of a second testing scheme, a set of *S-images* has been iteratively

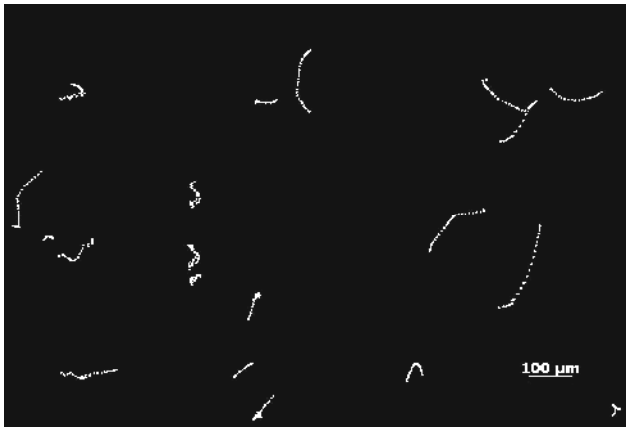


Fig. 3. Binary reconstruction of a time-lapse stack of 40 segmented and OR$^{ed}$ *S-images*.

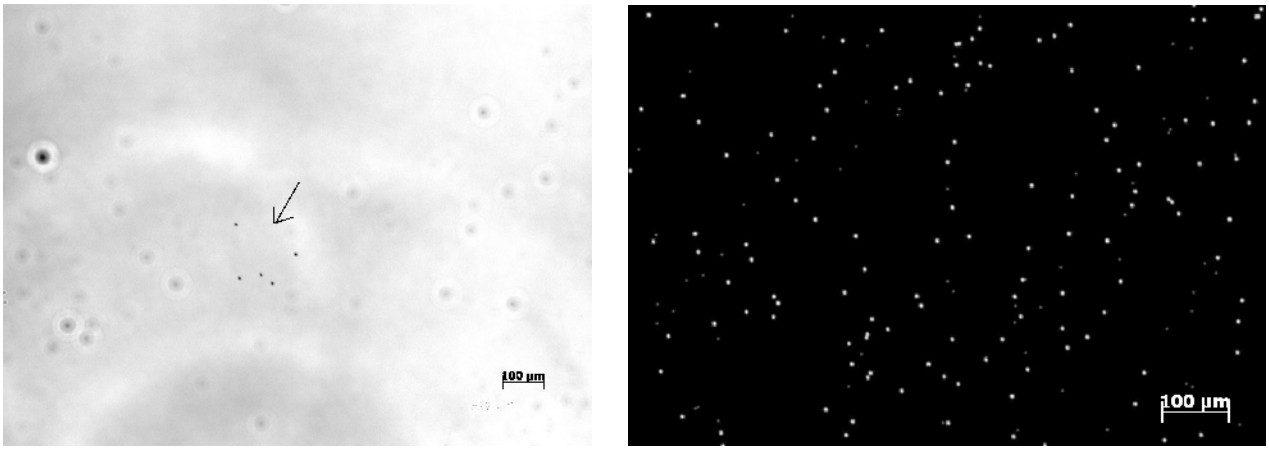

Fig. 4. BF and DF simulated images containing respectively 5 and 200 bacteria agglomerations.

generated with optical cell concentrations varying from 10 to 5000 cells/mm$^2$. Cells were to be chosen randomly in real-time from a set of 5 predefined models. This set was then used as input for the Matlab version of the IDL particle tracker described in [8] and results are shown in Table I. The Tracking Error (TE) is based on measurements metrics and is defined by:

$$TE = \frac{1}{N + E_2}\left(\sum_{n=0}^{N}\left(E_{1,n}\right) + E_2\right) \qquad (9)$$

where $N$ is the number of detected tracks, $E_1$ is the absolute error in the track length and $E_2$ is the absolute error in the track total number. Such a metric-based evaluation is precise and does not introduce evaluation errors. The tracking registration seems to be based on the nearest-neighbor algorithm, which explains the relative degradation of the performance with the rise of the cell concentration. Computation becomes exhaustive for very high densities and the algorithm fails.

The developed random track generator is parametrical and can be tuned to favor certain types of motility. Fig.3 shows an example of random two-dimensional tracks of bacteria under the 10x microscope objective and transmitted light dark field illumination. The image consists of a binary reconstruction of a time-lapse stack of segmented and OR[ed] *S-images*.

## IV. Conclusion

In this paper we have designed and implemented a general software simulator for optical microscope images. Such a simulator is of importance and in some cases a necessity for the evaluation of optical cell-tracking algorithms. After testing, the architecture has proven to be versatile. The simulator has yielded satisfactory images with different microscopy setups. Execution is also satisfactory especially as simulated images of quality highly comparable to real CCD acquired images have been generated. A popular particle tracking software has been evaluated and its performance in relation to cell concentration in the sample exposed.

In the future, the *SIS* will be re-implemented for more efficient execution and the testing framework will be extended to formal comparison of major cell-tracking algorithms. Furthermore, as cell motility data increases and cell-moving patterns characterized, *Tracking Generators* incorporating such patterns will be implemented. Cell movement in the simulations will be highly realistic providing an even more powerful scheme for ground-truth validation of tracking algorithms.

## References

[1] M. Wu, J. W. Roberts and M. Buckley, "Three-dimensional fluorescent particle tracking at micron-scale using a single camera", *Experiments in Fluids*, vol. 38, pp. 461-465, December 2004.

[2] H. Oku, N. Ogawa, M. Ishikawa and K. Hashimoto, "Two-dimensional tracking of a motile micro-organism allowing high-resolution observation with various imaging techniques", Rev. *Sci. Instrum*., vol. 76, 034301, 2005.

[3] A. Lehmussola, J. Selinummi, P. Ruusuvuori, A. Niemisto and O. Yli-Harja, "Simulating fluorescent microscope images of cell populations", *Proc. IEEE Engineering in Medicine and Biology*, September 2005.

[4] G. Rabut and J.Ellenberg, "Automatic real-time three-dimensional cell tracking by fluorescence microscopy", *Journal of Microscopy*, vol. 216, pp. 131-137, July 2004.

[5] K. Miura, "Tracking movement in cell biology", *Advances in Biochemical Engineering/Biotechnology*, vol. 95, pp. 267-295, 2005.

[6] J. Verest and D. Chetverikov, "Experimental comparative evaluation of feature point tracking algorithms", *Proc. Workshop on Eval. And Validation of Computer Vision Algos*, pp 183-194, 2000

[7] R. H. Luchsinger, B. Bergersen and J. G. Mitchell, "Bacterial swimming strategies and turbulence", *Biophysical Journal*, vol. 77, pp. 2377-2386, November 1999

[8] D. Blair and E, Dufresne, "The Matlab particle tracking code repository", http://www.deas.harvard.edu/projects/weitzlab/matlab/

[9] M. K. Cheezum, W. F. Walker and W. H. Guilford, "Quantitative Comparison of Algorithms for Tracking Single Fluorescent Particles", *Biophysical Journal*, vol. 81, pp.2378-2388, October 2001