TraCCA - A Complex Cellular Automata based Particle Tracking Framework

Davide Spataro, Paola Arcuri, Alessio De Rango,
William Spataro and Donato D'Ambrosio
Department of Mathematics and Computer Science
University of Calabria
Rende, Italy
Email: d.spataro@mat.unical.it

Alice Mari
Institute for BioEngineering
University of Edinburgh
Kings Buildings, EH9 3BF Edinburgh, UK
Email: alicemari45@gmail.com

***Abstract*—Particle tracking plays an important role in numerous fields of science for the investigation of the movement of sub-micron particles, micropsheres and molecules under microscopic observation. In this paper we present an algorithm for detecting and tracking particles based on geometrical difference evaluation and centroid displacement analysis to reconstruct the trajectories. This method works for n-dimensional input data provided that particles are represented by at least a centroid space coordinate and a geometrical entity which describe their shape. Since 2-D images are a common source of such data, we also present a framework for image-manipulation based on Extended Cellular Automata (XCA). We have applied and validated TraCCA in investigating the motility of *B. subtilis.* injected in a microfluidic device using 4100 images taken at 100 frames per second. Results show that the framework is able to reconstruct the trajectories as computed motion parameters that are in accordance with the ones found in literature. Eventually, a preliminary parallel version implemented on GPUs and distributed memory machines have produced promising scalability and speedup, proving that the methodology can be also applied for large datasets.**

***Keywords*—cellular automata, tracking, image processing, bacteria motility**

I. Introduction

Studying of the movement of sub-micron particles, microspheres and molecules under microscopic observation often requires their time trajectories from which important kinematic and dynamic properties can be computed. Several studies employ time-lapse microscopy, especially in the field of biophysics, as a tool to gather data and retrieve single particle time trajectories. The researcher usually relies on manual or semi-manual/interactive software to study such properties. However, this approach is unfeasible when the number of cells involved in the analysis is high.

In this paper we present an algorithm for detecting and tracking particles that is based on image processing techniques and to shape difference and centroid displacement analysis to reconstruct the trajectories. This method works for n-dimensional input data provided that particles are represented by at least a centroid space coordinate and a geometrical entity which describe its shape. Since 2D images are a common source of such data we also present framework for image-manipulation based on the Extended Cellular Automata(XCA) paradigm .

TraCCA has been successfully applied for the investigation of the motility of *B. subtilis.* injected in a micro-fluidic device using 4100 images taken at 100 frames per second, as reported in Section IV.

The paper is organized as follows: Sections II and III outline the proposed tracking algorithm and cellular automata based image processing framework, respectively; Section IV shows a detailed application of TraCCA referred to a real case study regarding bacterial motility, while conclusions and possible future works are reported in Section VI.

II. Tracking Algorithm

The objective of the tracking algorithm is to produce a set $T_n = \{t_i\}$ s.t. $t_i = \{c_k^i, c_{k+1}^i, \ldots, c_l^i\}$ of trajectories each described as a time-ordered list of positions in space from a set of input particles $P = P_1 \cup P_2 \cup \ldots \cup P_n$ and a function $\mathcal{D} : P \times P \mapsto \mathbb{R}$, the *distance* function. $P_i = \{p_i^j \mid 1 \leq j\}$ indicates all particles at time i and each particle p_i^j is defined by a centroid position, and a bounding box which describes its geometrical properties. $\mathcal{D}(p, q)$ measures the likeliness that a particle p has been transformed into q as a result of the application of a number of geometrical transformations such as translation, scaling, shearing or rotation (see Eq. 2). Indexes k and l, $k \leq l$ indicate the trajectory starting and ending time of the tracked movement respectively, and the length $l-k+1$ is its duration in time. Note that particles may appear or disappear at any time and hence $k \geq 0$ and $l \leq n$. Moreover, $|t_i| \leq l-k+1$ since a particle which has been successfully tracked from P_k to P_{k^*} can disappear for a certain time and may appear again in $P_{\bar{k}}$, $k \leq k^* \leq \bar{k}$. We only allow disappearing time $\bar{k} - k^* \leq \xi$ where $\xi \geq 0$ is a parameter of the algorithm. Each trajectory $t_* = \{c_k^*, c_{k+1}^*, \ldots, c_l^*\}$ is composed by positions of particles $p_k^{j_1}, p_{k+1}^{j_2}, \ldots, p_l^{j^*}$ at different times. This means that, for our purpose, particle $p_k^{j_1}$ at frame k has moved from c_k^* to location of particle $p_{k+1}^{j_2}$,c_{k+1}^* at time $k+1$ and to location of $p_l^{j^*}$, c_l^* at time l.

As an example, let us consider a human tracking system where each P_i could correspond to all the detected bodies in a video frame i and the distance function a linear combination of the euclidean distance between two detected bodies centroids and pixel-by-pixel difference in colors of all the pixels within their bounding boxes. In this context, it would make sense to

consider a not null disappearing time since it is not uncommon for the human detection module (which is in charge of producing the centroids and bounding box from the images) to skip recognizing a specific target only for a limited number of frames.

In order to construct the trajectories, the algorithm works sequentially from frame 1 to n processing, at each step, two subsets of particles, M_i and P_i, where M_i contains all the corresponding trajectories ending particles p_l^j that can still be expanded, i.e. $i - l \leq \xi$. Informally, the algorithm tries to augment an element in M_i using a particle in P_i making sure at most one particle is added to it, the same particle does not augment two different trajectories and the augmenting is performed s.t. the distance function is minimized.

Since at each step of the process a possible assignment between an element of M_i and one of P_i is sought, the algorithm can be thought to be similar to the *assignment problem* [5] and more specifically, it consists in finding a minimum weight matching (not necessarily perfect) in a weighted bipartite directed graph $G = (V, E)$ where $V = M_i \cup P_i$ is the set of nodes and M_i, P_i are the two partitions, $E = M_i \times P_i$ s.t. $e \in E, \mathcal{D}(e) \in \mathbb{R}$ is the weight of the edge. A valid matching $S \subseteq E$ must satisfy the following:

$$\forall (u,v) \in S \begin{cases} (w,x) \in S,\ v = x \iff u = w \\ \mathcal{D}(u,v) = \min_{x \in V_2} \mathcal{D}(u,x) \\ \nexists\, (w,v) \in E \text{ s.t. } \mathcal{D}(w,v) < \mathcal{D}(u,v) \end{cases} \quad (1)$$

If we denote the matching operator as the following recurrence relation $M_i \Diamond P_{i+1} = (T_{i+1}, M_{i+1})$, $M_0 = P_0$, then the tracking algorithm can be summarized as $(T_n, M_n) = M_{n-1} \Diamond P_n = (((P_0 \Diamond P_1) \Diamond P_2) \Diamond \ldots \Diamond P_n)$. If after the $\Diamond$ application a particle $p^* \in M_i \cup P_i$ remains unmatched[1], two cases have to be considered (see Figure 1):

1) if $p^* \in M_i$, the disappearing time counter u_{p^*} for p^* is updated and if $u_{p^*} > \xi$, p^* is not included in M_{i+1} and the corresponding tracked trajectory is flushed into T_i, otherwise it is retained. This handles the case when particles may disappear from the dataset for a number of time steps and appear again.
2) if $p^* \in P_i$, it corresponds to a newly appeared particle which is then inserted into M_{i+1}.

The pseudo-code reported in Algorithm 1 shows how the matching procedure is implemented. Note that the NEIGH procedure filters the possible candidates for a particle only to those which the euclidean spatial distance is less than a threshold parameter. In many real life applications particle displacement between two subsequent time-steps are small, so it would be useless trying to match a particle at time i with one at time $i + 1$ which are spatially far apart.

III. Manipulating images using XCA

In this section we present a Cellular Automata based framework for manipulating images that allows seamless parallel filters application.

[1] $\nexists\, (u,v) \in S$ s.t. $u = p^* \vee v = p^*$ (cf. Equation 1)

Algorithm 1: ssdsdsvc

```
1  Function MATCH (M_i, P_{i+1});
   Input  : Matched and to be matched particles M and P
            respectively.
   Output: (T_{i+1}, M_{i+1}), which are the updated set of
            trajectories and matched particles.
2  T_{i+1} = T_i
3  foreach p ∈ M_i do
4      neighbours[p] ← NEIGH(p, P_{i+1}) ;
5      foreach n ∈ neighbours[p] do
6          d[p][n] ← DISTANCE(p, n);
7      end
8      SORTBY(neighbours[p], d[p]);
9  end
10 foreach p ∈ M_i do
11     if neighbors[p].size ≠ 0 then
12         candidate ← neighbors[p].first
13     else
14         p.u ← p.u + 1;
15         continue;
16     end
17     if match[candidate] = NIL then
18         match[candidate] ← p;
19         p.u ← 0;
20     else
21         p' ← match[candidate]
22         if d[p][candidate] >= d[p'][candidate] then
23             neighbors[p].pop()
24         else
25             match[candidate] ← p;
26             neighbors[p'].pop() ;
27             p ← p';
28         end
29         go to 10;
30     end
31 end
32 foreach p ∈ P_{i+1} do
33     if match[p] = NIL then
34         T_i + 1.push(< p >)
35     else
36         FIND_TRAJ(T_{i+1}, match[p]).enqueue(p);
37     end
38 end
39 foreach p ∈ M_i ∪ P_{i+1} do
40     if p.u < P_r then
41         M_{i+1}.push(p)
42     end
43 end
44 return (T_{i+1}, M_{i+1});
```

A. Cellular Automata

Cellular Automata [15] are parallel computing models, whose evolution is defined by local rules. A cellular automaton can be thought as a d-dimensional space, called *cellular space*, subdivided in regular cells of uniform shape and size. Each cell embeds a *finite automaton*, one of the most simple and well known computational models, which can assume a finite number of states. At time $t = 0$, cells are in arbitrary states and the CA evolves step by step by changing the states of the

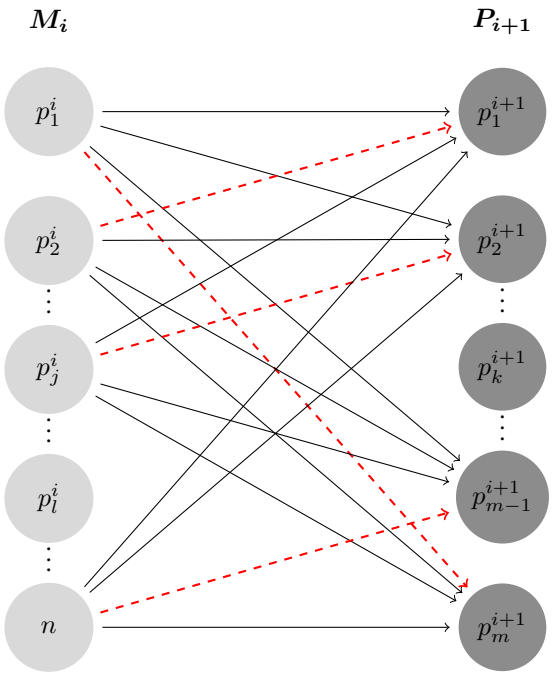

Fig. 1. An application of the $\Diamond$ operator referred to a frame P_{i+1} of m particles. Dashed red arrows highlight the solution S. For instance, the trajectory $t_* = p_a^s \rightsquigarrow p_1^i$ is lengthened by p_m^{i+1} becoming $t = p_a^s \rightsquigarrow p_1^i \rightarrow p_m^{i+1}$. Unmatching (no incident dashed red arrow are present) particle p_k^{i+1} causes a trajectory of length one to be created, while unmatching particle p_l^i causes its disappearing time counter to be updated and the corrensponding trajectory to be possibly finalized (if $u_{p_l^i} > \xi$). Note that, for the sake clarity, arrows for the unmatched nodes are omitted.

cells at discrete time steps, by applying the same local rule of evolution, i.e. the cell's *transition function*, simultaneously (i.e. in parallel) to each cell of the CA. Input for the cell is given by the states of a predefined (usually small) set of neighboring cells, which is assumed invariant in space and time.

Extended Cellular Automata ([13], [14]) represents an extension of the original CA computational paradigm. The main differences between XCA and classical CA are that the CA state can be expressed as Cartesian product of $n > 0$ *substates*, the transition function can also be decomposed into *elementary processes* that can be parametrized, and non-local operations, that go under the name of *global functions* are allowed. The initial conditions of the system are obtained by preliminary applying a non-local initialization operation.

B. Definition and Usage

The XCA adopted for manipulating images is defined as a 7-tuple $IFCA = \langle R, X, Q, P, \sigma, \Gamma, \gamma \rangle$ where:

- R is the 2-dimensional cellular space.
- $\Gamma \subseteq R$ is the region over the global operation are applied
- $X = X(x_0, y_0) = \{(x, y) : |x - x_0| \leq r \wedge |y - y_0| \leq r\}$ defines the *Moore*'s neighborhood relationship of radius r
- $Q = Q_r \times Q_g \times Q_b \times Q_a$ representing the pixel color channels in the RGBA space.
- P is the set of global parameters
- ψ is the initialization function (which is in charge of reading images from files and converting them to substates).
- $\sigma = \{\sigma_i : Q^{|X|} \mapsto Q\}$ is the set of image convolutional filters (corresponding to XCA elementary processes)
- $\gamma = \{\gamma_i : Q^{|\Gamma|} \rightarrow Q^{|\Gamma|}\}$ is the set of non-local filters (the XCA global functions).

In order to take advantage of the intrinsic parallel nature of CA, IFCA is implemented augmenting an existing CA library, **OpenCAL** [6] [7] which is empowered with a set of procedures that allows seamless input/output and image convolution. More specifically each image is represented as an cellular automaton whose substates represent the color of the pixel and filters implemented as elementary processes composition (see listing 1).

```
//Model and CA engine                                                    1
CALMooreNeighborhood<DIM,RADIUS> neighbor;                               2
MODELTYPE calmodel(IMG_SIZE,&neighbor,SPACE_FLAT);                       3
CALRun calrun(&calmodel, 1, STEPS, UPDATE_IMPLICIT);                     4
CALSUBSTATE* bgr=calmodel.addSubstate<PIXELTYPE>();                      5
//Image loading                                                          6
bgr->loadSubstate(*(new std::function(loadImage<                         7
    PIXELTYPE>)), "image.tiff");
//Image Filters Creations                                                8
ContrastStrchFlt<...>csf(bgr,1285,1542,0,65535,1);                       9
ThresholdFlt<...> tf (bgr,0,61680,0,65535);                              10
calmodel.addElementaryProcess(&cst);                                     11
calmodel.addElementaryProcess(&tf);                                      12
//Trigger execution                                                      13
calrun->run();                                                           14
```

Listing 1. Example of usage of the IFCA XCA engine. Lines 1-5 create a model and runtime for an image of size IMG_SIZE and dimension DIM. Lines 7 loads the image into the substate which, for the sake of brevity, is unique since the input image is assumed to be single channel. Lines 9-10 show the creation of two user defined filters and Lines 11-12 load them into the engine. The order in which they appear in the code specifies the order of execution that is triggered at line 14.

IFCA is extensible as it allows the application of user-defined filters by only specifying the pixel transformation rules (or kernels in case of convolutional ones). Filters are then executed by the IFCA engine which comes with two parallel (OpenMP and OpenCL) and a serial implementations. The parallel execution is completely transparent and automatic and may be extremely useful in such cases where the size of the image is large or filters are particularly computationally demanding.

IV. Motility Analysis of B. subtilis

A. Introduction

In this section we present an application of TraCCA to the analysis of motion of *B. subtilis*. The analysis of trajectories in bacteria is really interesting because it is a non-invasive way of extracting much information about their chemotaxis which is the ability of bacteria to sense and respond to chemicals. Thanks to chemotaxis, bacteria are able to reach a source of nutrient and move away from repellents, which makes it a key characteristic for their survival. Studying chemotaxis is really interesting not only because it has a strong influence on many

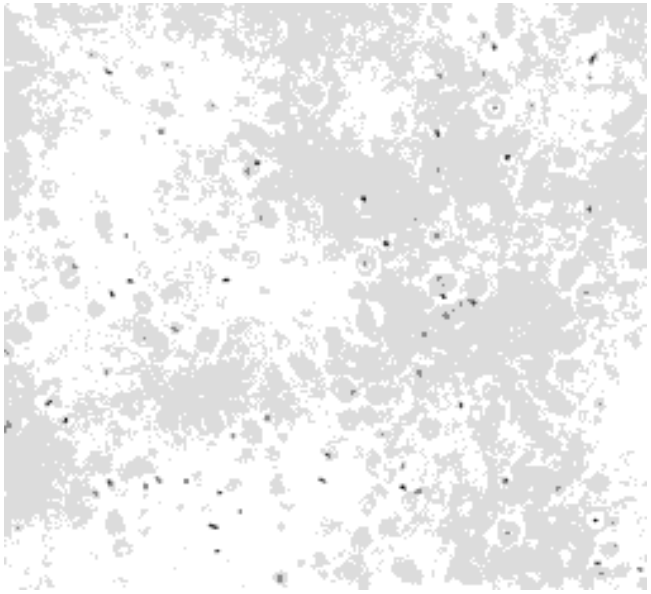

Fig. 2. Origianal frame. The bacteria are the small darker cluter. Light gray and white is noise and background.

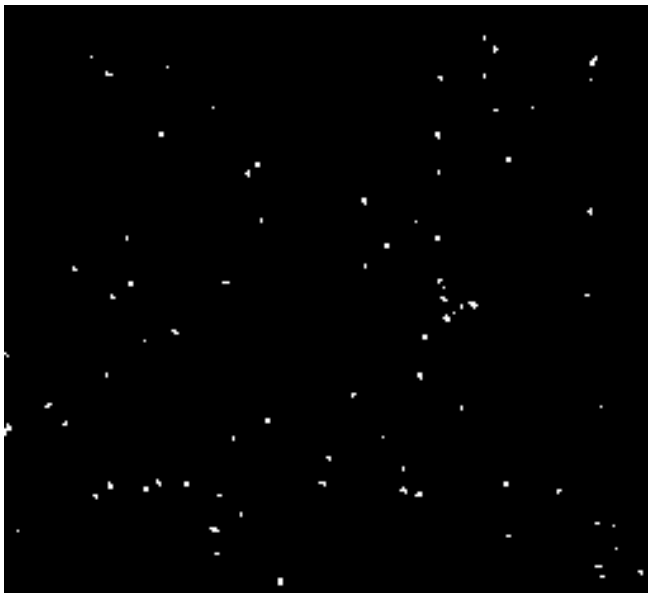

Fig. 3. This represents a segmented and inverted version of the image on the left. White cluster are bacteria.

biological mechanisms, e.g. biofilm formation and disease pathogenesis but also because evolution's natural selection has optimized bacterial chemotaxis making bacteria excellent source-seekers and their strategies can be used to design bio-inspired, simple and efficient algorithms for robotic source locating systems. The motility of *B. subtilis* is composed of a series of *run* (bacteria swim along smooth segments) and *"tumbles"* (cells stop and randomly select a new direction). Combining run and tumbles bacteria are able to direct their motility in order to reach nutrients and move away from repellent, in other words, to do their chemotaxis. An algorithm that is able to track bacteria is really useful because from bacterial trajectories much information about the way bacteria perform chemotaxis (e.g. duration of the run, swimming speed, frequency of tumbling events) can be extracted. This information is important in studying the strategy they adopt to reach a source of nutrient; moreover it is also interesting to investigating how the trajectories changes as certain environmental conditions change, such as temperature, oxygen concentration or gradient of chemicals.

In this work we used *B. subtilis* strain `OI1085` for the tracking experiment. Cells were taken from frozen stock, resuspended in CAM (Cap Assay Minimal) and shaken (37°, 100 rpm) until the optical density $OD600 = 0.3$ was reached; we then diluted the suspension 1:10 in CAM. The bacterial suspension was injected in a micro-fluidic device. The micro-fluidic device is a simple device made by *PDMS* and glass, composed of three parallel channels (height $100\,\mu\mathrm{m}$). In the central channel ($600\,\mu\mathrm{m}$ wide) bacteria were hosted and observed. Two walls of *PDMS* ($200\,\mu\mathrm{m}$) separate the central channel from the left and right channels were oxygen was flown in order to reach a concentration of oxygen closed to 100% in the central channel. 10 minutes after the injection of bacterial suspension a video was acquired at 100 frames per second for $41\,\mathrm{s}$ (#4100 frames). All images were acquired through a $10\times$ phase contrast objective (*Nikon* microscope), using binning 2×2. All images are 512×512 pixels and have been exported in 16-bit grayscale `TIFF` image format.

B. Bacteria Segmentation

The *B. subtilis* cells typically have a large range of motion patterns and the cell soma generally appears as a dark area surrounded with a white halo. Colors can be inverted and the cell may appear white when it moves out of focus sufficiently (see Figure 2). In order to automatically detect *subtilis* cells, it is possible to use a histogram-based thresholding method as suggested in [1]. A key step of tracking bacteria is to individuate first, and then to label and describe each of the bacteria present in the images. For this purpose, a segmentation preprocessing phase is performed on all the images. Segmentation is carried out by means of a threshold method [2] which produces a bipartition of pixels based on the color intensity. The value of pixel (x, y) in image g is given by the following, where P is a predicate and f is the original image:

$$g(x,y) = \begin{cases} 1, & \text{if } P(f(x,y), T) \\ 0, & \text{otherwise} \end{cases}$$

As a consequence, a drawback of using the threshold method is that, among all inter-pixels relationships, color intensity difference is the only involved by the bi-partition process. This can easily lead to binary regions where pixels are not contiguous or to miss or include relevant or unwanted pixels respectively, with these effects getting worse as the noise increases. In the considered case, however, the threshold method works well because after the application of the `contrast stretch filter` the analysed images present high contrast between the background and cells soma (see Figure 2), since the filter stretches or scales the range of pixel values between an upper and lower limit. More specifically, color values that are above or below this range are saturated to the upper or lower limit values respectively, while values that lay in the interval are

scaled according to the following formula:

$$g(x,y) = \begin{cases} L_o & \text{if } f(x,y) < L_i \\ H_o & \text{if } f(x,y) < H_i \\ L_o + (f(x,y) - L_i)\frac{H_o - L_o}{H_i - L_i} & \text{if } L_i \leq f(x,y) \leq H_i \end{cases}$$

where $[L_i, H_i]$ defines the interval in the original image which is linearly scaled into the interval $[L_o, H_o]$.

Eventually, all the images go through a noise reduction stage which employs a combinations of *gaussian*, *laplacian* and *blurring* filters [3]. It is worth to note that filters parameters are dataset dependent and the best set of values experimentally determined.

C. Bacteria Tracking

Binary images are then used to extract the relevant information (shape and centroid) for each of the segmented bacteria. This is done by interpreting each image as a graph whose nodes are pixels and arc (i, j) exists if pixels i and j are neighbors (according to *Moore*'s relationship, see section III-A). A bacterium corresponds to a connected component that can be easily individuated by using the DFS visiting algorithm. Once all pixels that make up the cell soma are individuated, a unique *ID* id_i which identifies the bacterium uniquely, a centroid c_i and a bounding box s_i which describe describes the location, the shape and the area, respectively, are computed for each bacterium i. All the bacteria whose extension is less than $M_e = 2px$ are ignored and no further considered. The creation and manipulation of all the geometrical entities involved in this latter phase are carried out by means of the computational geometry library `CGAL` [4].

In order for the application of the algorithm described in Section II, a distance function is required. Among possible functions, a linear combination of the centroids displacement and shapes difference was adopted:

$$D(c_i, c_j) = a\sqrt{(x_{c_i} - x_{c_j})^2 + (y_{c_i} - y_{c_j})^2} + b|S_i \cap S_j| \quad (2)$$

where S_i and S_j are the sets of pixels within the boundaries of the bacteria's bounding box. Best values of weights a and b were experimentally determined in being 0.6 and 0.4 respectively.

V. Analysis and Validation

Starting from the relationship $D_b = \frac{v^2 t_t}{2}$ [**?**] that links the diffusion coefficient to both swim velocity (v) and tumbling time (tt), we have partitioned each trajectory into running and tumbling events. These events allow the swimming speed to be calculated as difference between subsequent bacteria positions and average tumbling time as the overall time spent tumbling over by the number of tumbling events. Inspired by the work of *Wong-Ng et al.* [10] and *B. Masson et al.* [11] an algorithm, implemented in *MATLAB*, was adopted for detecting such running/tumbling events (see Figure 4) which in turn were used to extract all the relevant bacterial motility parameter from the tracked trajectories[2].

[2]Tumbles are associated to a decrease in advancing velocity and an abrupt change in the angular velocity (i.e. in the direction of the motion). Fixing a threshold on both advancing and angular velocities it is possible to identify running/tumbling and use them to computes the mean values t_t and v over all the trajectories of the experiment.

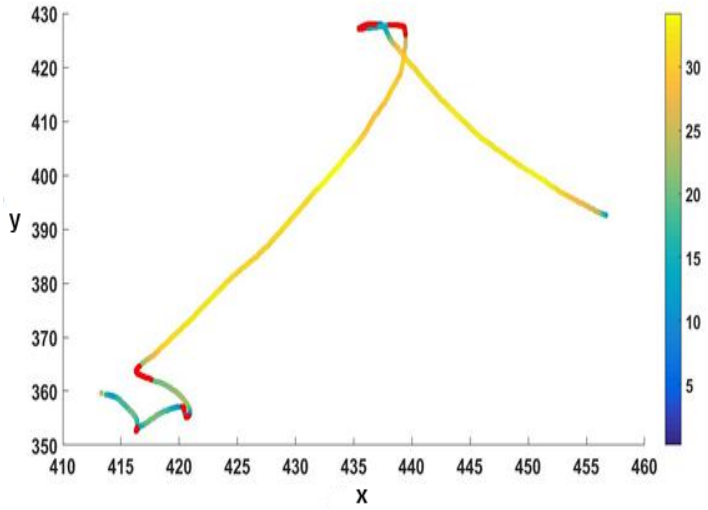

Fig. 4. This figure depicts the tracked trajectory of a single bacterium. Blue-yellow colors gradient shows the velocity. The red segments represent the identifies tumbles.

A mean swimming velocity of $18\,\mu\mathrm{m\,s^{-1}}$, a mean run time (time spent swimming straight) of $0.8\,\mathrm{s}$ and a tumble time of $0.18\,\mathrm{s}$ were found. All these values[3] are in accordance with the results of *Cisneros et al.* [12].

VI. Conclusion

This premiliary work we presented a an cellular automata based framework for tracking centroid-bounding box represented particles. We also presented an application of the proposed framework on tracking the motion of *B. subtilis* in a microfliudic device in order to retrieve the average swimming velocity, running and tumble times. The results are in accordance with [12].

A. Future works

Due to the possible large number of particles and frames involved in such analysis a parallel GPGPU+MPI version of the framework is being developed and it will be applied to the analisys of a much larger dataset to test its scalability and speedup. It is also important to note that under the assumption of associativity of the *assignment operator* $\Diamond$ the tracking algorithm could be implemented using the design pattern of *parallel reduction*.

Acknowledgment

The authors would like to thank NVIDIA for providing GPUs and Prof. Filippo Melascina from the University of Edinburgh, Institute of BioEngineering, UK who kindly gave access to the microscopy equipment.

[3]These motility parameters were also calculated using the *Mean Square Displacement* (`MSD`) in time (which for the sake of brevity is only outlined here) and considering the following relationship $MSD(t) = \frac{1}{2}\frac{v^2 t_R^2}{2t/t_R + e^{\frac{-2t}{t_R}} - 1}$ [**?**] where v is the swimming speed, t_R is the timescale associated to rotational diffusion that is $t_R = 2t_t$, it is possible to obtain t_R and v via fitting. t_t and v are then used to compute D_b.

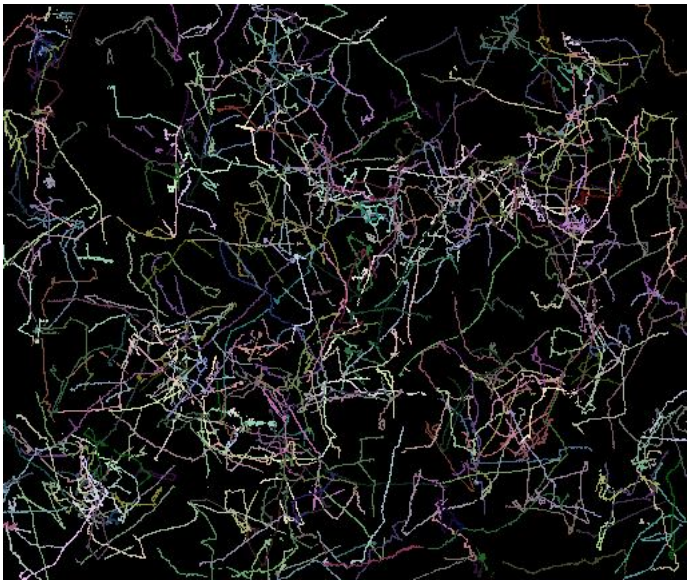

Fig. 5. This figures shows the collective view of all the tracked bacteria tracjectories. A random color is associated to each trajectory.

REFERENCES

[1] S. Cho, R. Haralick, and S. Yi, "Improvement of Kittler and Illingworths minimum error thresholding," Pattern Recognit., vol. 22, no. 5, pp. 609 617, 1989.

[2] Shapiro, Linda G, Stockman, George C. (2002). "Computer Vision". Prentice Hall. ISBN 0-13-030796-3

[3] G. Deng and L. W. Cahill, An adaptive Gaussian filter for noise reduction and edge detection, Nuclear Science Symposium and Medical Imaging Conference, 1993. 1615-1619 vol.3, 10.1109/NSSMIC.1993.373563

[4] CGAL, Computational Geometry Algorithms Library, http://www.cgal.org

[5] Bellman, R. E. Review: Eugene L. Lawler, Combinatorial optimization: networks and matroids . Bull. Amer. Math. Soc. 84 (1978)

[6] OPENCAL download page: https://github.com/OpenCALTeam/opencal/releases

[7] OPENCAL download page: https://github.com/OpenCALTeam/opencal-user-guide

[8] Nadine Tarantino, Jean-Yves Tinevez, Elizabeth Faris Crowell, Bertrand Boisson, Ricardo Henriques, Musa Mhlanga, Fabrice Agou, Alain Isral, and Emmanuel Laplantine. TNF and IL-1 exhibit distinct ubiquitin requirements for inducing NEMO-IKK supramolecular structures. J Cell Biol (2014) vol. 204 (2) pp. 231-45

[9] http://tinevez.github.io/msdanalyzer/

[10] Wong-Ng J, Melbinger A, Celani A, Vergassola M (2016) The Role of Adaptation in Bacterial Speed Races. PLoS Comput Biol 12(6): e1004974. doi: 10.1371/journal.pcbi.1004974

[11] Noninvasive inference of the molecular chemotactic response using bacterial trajectories Jean-Baptiste Masson, Guillaume Voisinnea, Jerome Wong-Nga, Antonio Celania, and Massimo Vergassola, 2011

[12] Cisneros, L. H., Kessler, J. O., Ganguly, S., Goldstein, R. E. (2011). Dynamics of swimming bacteria: Transition to directional order at high concentration. Physical Review E, 83(6), 061907.

[13] Salvatore Di Gregorio, Roberto Serra, An empirical method for modelling and simulating some complex macroscopic phenomena by cellular automata, Future Generation Computer Systems, Volume 16, Issues 23, December 1999, pp. 259-271.

[14] Spataro, Davide, et al. "The new SCIARA-fv3 numerical model and acceleration by GPGPU strategies." International Journal of High Performance Computing Applications (2015)

[15] John Von Neumann. 1966. Theory of Self-Reproducing Automata. Arthur W. Burks (Ed.). University of Illinois Press, Champaign, IL, USA