31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

Streaming Level Set Algorithm for 3D Segmentation of Confocal Microscopy Images

Alexandre Gouaillard, Kishore Mosaliganti, Arnaud Gelas, Lydie Souhait, Nikolaus Obholzer and Sean Megason

***Abstract*— We present a high performance variant of the popular geodesic active contours which are used for splitting cell clusters in microscopy images. Previously, we implemented a linear pipelined version that incorporates as many cues as possible into developing a suitable level-set speed function so that an evolving contour exactly segments a cell/nuclei blob. We use image gradients, distance maps, multiple channel information and a shape model to drive the evolution. We also developed a dedicated seeding strategy that uses the spatial coherency of the data to generate an over complete set of seeds along with a quality metric which is further used to sort out which seed should be used for a given cell. However, the computational performance of any level-set methodology is quite poor when applied to thousands of 3D data-sets each containing thousands of cells. Those data-sets are common in confocal microscopy. In this work, we explore methods to stream the algorithm in shared memory, multi-core environments. By partitioning the input and output using spatial data structures we insure the spatial coherency needed by our seeding algorithm as well as improve drastically the speed without memory overhead. Our results show speed-ups up to a factor of six.**

I. Introduction

Researchers in embryogenesis and oncology, among others, rely on automated segmentation of cells to understand the complex processes of tissue morphogenesis. In our work, we are attempting to recover entire lineages of zebrafish cells during its embryonic development. From a zygotic single cell stage, the embryo rapidly multiplies over a period 2 days to become a multi-cellular organism with millions of cells. Using different fluorescent markers to stain the nucleus and the membrane, we obtain large $3D+t$ multichannel datasets of the embryo using multiphoton microscopy. A single $3D$ dataset is acquired every 4 minutes and has pixel dimensions of $1024 \times 1024 \times 100$, pixel spacings of $0.2 \times 0.2 \times 1 \mu m^3$, file-size of $143MB$, and contain > 1500 detected cells. The lineage reconstruction involves segmenting (3D), tracking (3D+t) and classifying cells based on their position, shape, gene expression and observing their trajectories. From the image analysis stand-point, these tasks involve developing algorithms for cell profiling, counting, distribution statistics, shape analysis, segmentation and tracking of cells.

Cells/nuclei segmentation is often the first step in any quantitative analysis protocol. It involves uniquely identifying fluorescent marked cells and organelles, such as nuclei, that are spatially correlated but whose position, number, and geometry must be determined [1]. The problem is complicated by individual variations in intensity, geometry, relative orientation and overlapping boundaries (Fig. 1). In microscopy Images, the nuclei/cells which are the fundamental biological entities of interest often appear as overlapping or touching each other. Identifying each nucleus separately in a biologically consistent fashion is non-trivial. While some biochemical stains provide viable clues in the form of sharp color-space gradients at the boundaries, others exhibit a narrow *neck* at the site of overlap between two nuclei. We implement an approach that elegantly incorporates available cues and shapes into a viable segmentation pipeline [2]. We use geodesic active contours incorporating shape priors in a level-set framework to provide nuclei segmentations.

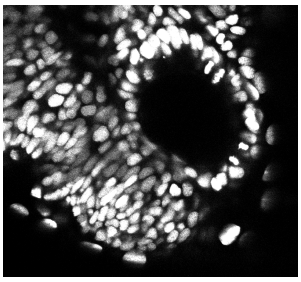


Fig. 1. Multiple cells in close contact and in the same field of view.

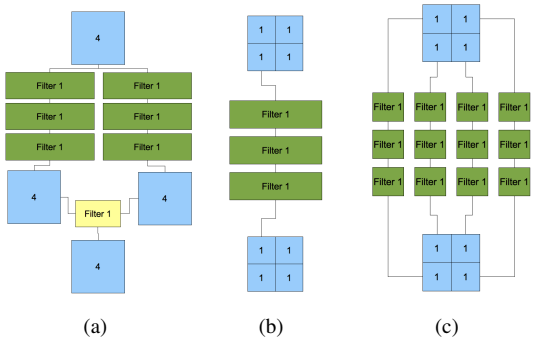
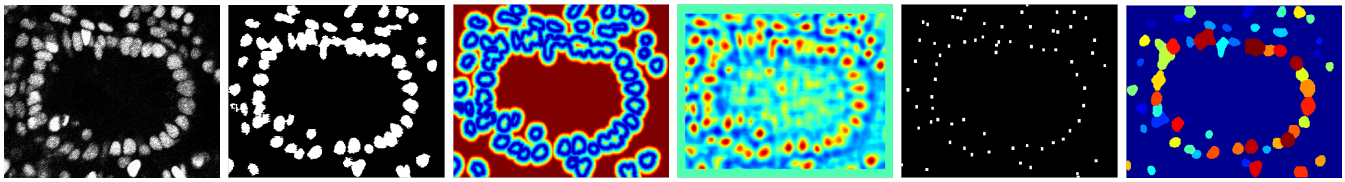


Fig. 3. (a) Multithreading with non separable output leads to higher memory consumption and additional processing to merge the temporary outputs from each thread into the final output. (b) Streaming allow to decrease the memory consumption by processing sub regions of the data at a time. (c) streaming and multithreading allow to process all data in parallel with no memory overhead compared to single threaded processing, but supposes that both input and output can be divided into independent sub-regions.

Systems Biology Department, Harvard Medical School, 200 Longwood Avenue, Boston MA 02215, USA. alexandre_gouaillard@hms.harvard.edu



(a) (b) (c) (d) (e) (f)

Fig. 2. (a) Confocal image showing nuclei of the zebrafish ear. (b) Extracted foreground (c) Gaussian correlation map showing convex regions and suppressing non-convex ones. (d) Distance map image. (e) Seed locations. (f) Level-set segmentations.

Figure 2 provides an example of our single-threaded level-set based cell segmentation algorithm. After filtering and adaptive thresholding operations, the nuclei foreground is detected in (b). The distance maps are then computed on the foreground in (c), and in (d), we detect convex image regions. By combining these image features in a linear manner, a speed image is computed for level-set evolution. In order to seed the level sets, we detect the maxima regions (seed points) of the distance map and the convexity map (e). The seed points are placed in a priority queue according to a quality factor. Starting from the seed with the highest priority, a level set function is initialized and evolved. Note that during evolution, a level set function may occupy other yet unprocessed seed points. This destroys any remaining (lower priority) seed points which are removed from the queue. Hence, we converge on a segmentation of the image by running through all the seeds in the queue as shown in (f).

Earlier, we mentioned that our large datasets contain thousands of cells. By representing each cell with a single level set function iteratively processing a single cell at a time is computational expensive. Since a cell occupies a small region of interest, it is possible to simultaneously process more than one cell by splitting the input images into several regions and combining the outputs in the end. furthermore our seeds priority queue design supposes that all the seeds corresponding to a spatial region are processed sequentially according to their priority. However, there are problems when cells lie across the splitting planes. Another significant factor to be considered is that the individual threads work load need to be balanced for maximum efficiency. The partioning needs to be sensitive to the asymmetric spatial distributions of cells. In this work, we explore methods to make the algorithm parallel and multi-threaded by partitioning the input and output using spatial data structures. Our results show speed-ups up to a factor 6 in some cases, and 2.5 in our case. We present in this paper all the results, as some reader might have slightly different algorithms or environments where the spatial constraint or memory constraint could be removed, and/or the memory constraint is stronger than in our case (we suppose the Image fits in memory), cases and environment for which we provide faster solutions as a side product of our research.

The rest of the paper is organized as follows. Section II describes related work in speeding up the level set methods. We define our complete level-set streaming solution in Section III. Results on segmenting the zebrafish ear from confocal images are presented in the same section. Finally, in Section IV, we provide a summary and describe our plans for the future.

II. Related Work

Multithreading refers to the process of running some code concurrently. This kind of process take advantages of multi-CPU or multi-core CPU (or both) machine that are ubiquitous nowadays. Each thread share the same memory space and can either exchange data or otherwise communicate through this shared memory. This is in sharp opposition with clustering or grid computing, that requires specific code to communicates between processes or nodes , adding an extra layer of complexity above the algorithm. In multithreading, concurrent reading to the same address is not prone to error, and reading operation are then thread-safe. However one must be extra careful when modifying a shared address as the result of concurrent modifications will depend on the order in which threads will have had access to the address. Solutions exist to lock the access to a shared memory address while one thread is writing to it, but it can lead to degradation of performance as other thread that would want to access the same address would just be suspended meanwhile. If it is a small variable and a quick write, the degradation can be acceptable. For image processing algorithm, locking the output image to write to it is an overkill.

All the image processing algorithms based on filtering using a kernel are intrinsically streamable (see figure 3:b and c, as each pixel of the output will be written only once and independently. Most of the problem then comes from the size of the kernel, that will require extension, or padding the region of the input used locally in each thread to be able to compute the output.

In our case, region of interest of cells can slightly overlap, and two overlapping ROI were to be processed in two different thread, we would have a race between the two thread, the last one writing in the output overwriting the first one. Moreover, the design of our seeding solution supposes spatial coherence within a thread. One possible solution would be to have a temporary output, of the same size as the final output, for each thread as illustrated in 3:a. But as we would like to run this algorithm in batch on a cluster, we face a limited memory environment, and we cannot afford the extra cost, especially as the images biologist acquire tend

to grow faster than any cluster node memory. We aim at a multithreaded, streamed solution as illustrated in Figure 3:c.

Most of the work on using level set for cell segmentation is relatively new [3]–[5]. We provided implementation of the corresponding algorithms for the ITK library [6], [7]. It does not seems that anyone have proposed multithreading or streaming versions of these algorithms.

III. Method and Results

We use a single time point of a dataset which contains 979 cells,as confirmed by both visual inspection and manual segmentation by biologists. Those 979 cells generate 1564 seeds. We tried different multithreading and streaming solutions on a windows XP 64 bits machine, with two quad core xeon processors. The normal memory consumption of the algorithm is roughly 600 MB.

The first row of the table I how the results of the original algorithm.

We first used a naive multithreading solution, as illustrated in 3:a. Each thread can read from the original, full, image from memory. Seeds are sequentially fetched by each thread and processed separately using the same original algorithm. Finally each thread as a temporary output where it can writes, allowing for the seeds optimization. It lead only to good speed enhancement but provided poor segmentation results. The second set of rows (separated by a line from the previous one), shows the results in this case. We can see that almost all the seeds are used, which leads to over segmentation. Because two seeds that correspond to the same cell can be processed by different thread independently, the priority queue design can not be used at its full potential. The speed improvement is obvious with a factor 6.8 when using 8 cores. It is counterbalanced by the high number of seeds processed, and the inferior quality of the segmentation which convinced us not to use this solution in our case. It would be one of our choice for an algorithm for which the memory is not an issue, and which wouldn't have spatial constraint on the seeds.

We then decided to enforce spatial consistency by dividing both input and output in regions of equal sizes as illustrated in 3:c. The third set of rows contains the corresponding results. We can see that we are converging toward the optimal number of processed seeds (979), but it does not translate in better performances. A the number of seeds in a given region can greatly vary from one region to the other, there is a great unbalance on the workload of each thread. Our experiments show that in average on our datasets 37% of the time is wasted waiting for other threads to finish. Additionally, cells that lie across the boundaries of the splitting planes are not well segmented as the level set evolution is constrained within the region to enforce non overlapping output region for each thread. This algorithm (with the solution proposed later in this chapter for the problem of the boundary cells) would be our solution of choice for algorithm that either are spatially independent, separable, and/or have memory limitation which would make streaming a must have. As all the partitions of the original input have the same size, the peek amount of memory is directly linked to the number of partitions and to the number of thread used. This solution allows for the design of an algorithm that would probe the amount of memory on a computer and adapt itself to this limit, transparently to the user.

Using binary spatial partitioning of the original image with respect to the seeds, we achieve both spatial coherence and load balancing. This method is perfect in a multithreaded and streamed environment where the number of threads and the number of partition are the same and where all the image fits in memory. The fourth set of rows contains the corresponding results. The results are really interesting, with a much better handling of false positive seeds (the number of seeds eventually processed are closer to 979), and a much better efficiency as less time is spend by threads waiting for others. Unfortunately, the quality of the result is still not at the level of the original algorithm, mainly because of the strong spatial constraint on the evolution of the level sets as illustrated on figure 4.

After computing regions using BSP we traverse the structure to check for the seeds that lie across the boundary, take them out of the multithreaded seed queues and put them in a special queue that will be processed separately. This queue will have access to the entire image and cannot be multithreaded, resulting in a slight performance loss. The fifth and lst set of rows illustrates the result. We made the difference between the seeds that where processed in the multithreaded queues and the boundary seeds processed in a single threaded queue. We can see that there is no performance increase above 4 threads, as the extra number of thread available is almost entirely compensated by the fast that there are many more splitting planes, and thus many more boundary cells. This point of inflection, the number of threads above which performance does not improve, is of course dependent on your data. Our data-sets represent very dense populations of cells. We believe other datasets, like drosophila datasets for example, could scale up better, as the cells are not so densely packed in 3D. This is our solution of choice nevertheless, as we have the same high quality as the original algorithm, more than twice faster (4 threads) with no memory overhead.

IV. Conclusion and Future Work

We provided here several multithreading solutions for different classes of image processing algorithms. Different hypothesis on input and output, as well as on the memory consumption and speed enhancement can guide to user to pick its solution of choice. We successfully designed a multithreaded version of our algorithm that provide us substantial speed improvement (x2.5) for the same quality of segmentation with no memory overhead.

In a future work, we would like to investigate clustering solution to run this algorithm on the 1,000+ time points of our current datasets, and the 10,000 time points that datasets produced by the end of the year may contain.

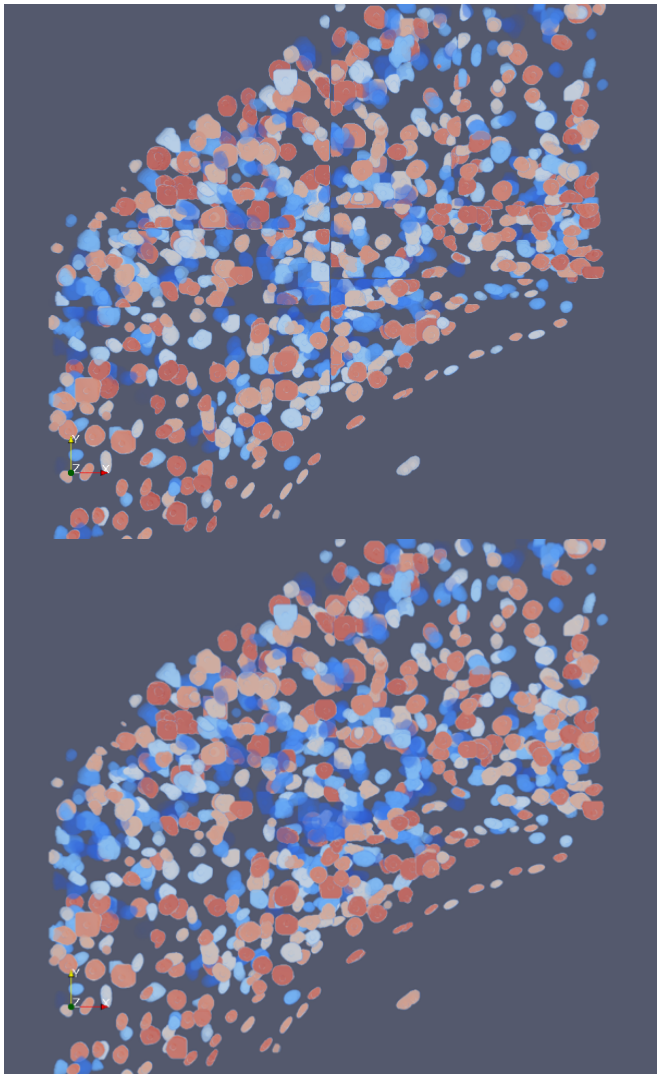


Fig. 4. Top: all level sets evolution is constrained inside the regions, which lead to clear under segmentation. Bottom: cells that lie across the boundaries are processed separately leading to complete segmentation of all cells.

Nb T	MT	Seeds	MT cells	ST	time (s)	cells per s	S %	Mem (GB)
1	0	P.Q.	-	979	1829	0.54	-	0.6
1	M	Rand	979	-	1833	0.53	-	0.6
2	M	Rand	1246	-	1226	1.02	91	0.7
4	M	Rand	1398	-	741	1.89	86	0.9
8	M	Rand	1488	-	438	3.40	80	1.4
1	S	-	979	-	1850	0.53	-	0.6
2	S	-	981	-	955	1.03	91	0.6
4	S	-	986	-	711	1.39	86	0.6
8	S	-	1011	-	430	2.35	80	0.6
1	S	BSP	979	-	1772	0.55	-	0.6
2	S	BSP	982	-	930	1.06	91	0.6
4	S	BSP	990	-	525	1.89	79	0.6
8	S	BSP	997	-	326	3.06	62	0.6
1	S	BSP+	979	0	1812	0.54	-	0.6
2	S	BSP+	883	96	998	0.98	82	0.6
4	S	BSP+	807	174	731	1.34	37	0.6
8	S	BSP+	699	284	716	1.37	2	0.6

TABLE I

TABLE OF RESULTS. FIRST COLUMN IS THE NUMBER OF THREADS USED. SECOND COLUMN IS THE SEEDING PARTITIONING POLICY, PRIORITY QUEUE (PQ), RANDOM SEQUENTIAL ACCESS TO THE QUEUE, SPATIAL DIVISION, BINARY SPATIAL PARTITION, AND HYBRID BINARY SPATIAL PARTITION WITH RESPECT FOR BOUNDARIES. THIRD AND FOURTH COLUMNS ARE, THE NUMBER OF CELLS MULTITHREADED, AND SINGLE THREADED, RESPECTIVELY. THE TOTAL RUNNING TIME, IN SECONDS, FOLLOWS. THE LAST THREE COLUMNS ARE THE THROUGHPUT RATE IN CELLS PER SECOND, THE EFFICIENCY OF DOUBLING THE NUMBER OF THREADS, AND THE MEMORY FOOTPRINT.

V. ACKNOWLEDGEMENTS

This work was funded by a grant from the NHGRI (P50HG004071-02) to found the Center for in toto genomic analysis of vertebrate development.

REFERENCES

[1] E. S. K. Khairy, E. Reynaud, "Detection of deformable objects in 3D images using markov chain monte carlo and spherical harmonics," in *MICCAI*, 2008, pp. 1083–1091.

[2] K. Mosaliganti, L. Cooper, R. Sharp, R. Machiraju, G. Leone, K. Huang, and J. Saltz, "Reconstruction of cellular biological structures from optical microscopy data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 863–876, 2008.

[3] A. Dufour, V. Shinin, S. Tajbakhsh, N. Guillen-Aghion, J. C. Olivo-Marin, and C. Zimmer, "Segmenting and tracking fluorescent cells in dynamic 3-d microscopy with coupled active surfaces," *Image Processing, IEEE Transactions on*, vol. 14, no. 9, pp. 1396–1410, 2005.

[4] T. Chan and L. Vese, "An active contour model without edges," in *Scale-Space Theories in Computer Vision*, 1999, pp. 141–151.

[5] L. Vese and T. Chan, "A multiphase level set framework for image segmentation using the mumford and shah model," *International Journal of Computer Vision*, vol. 50, pp. 271–293, 2002.

[6] K. Mosaliganti, B. Smith, A. Gelas, A. Gouaillard, and S. Megason, "Level set segmentation: Active contours without edges," *The Insight Journal*, 2008.

[7] ——, "Segmentation using coupled active surfaces," *The Insight Journal*, 2008.