Parallel Quad-Edge Active Contours for image segmentation

Daniel GONZALEZ[1], Lauriane ROHFRITSCH[1,2], Manon FAURE[1], Lydia DANGLOT[2],
Vannary MEAS-YEDID[1], Jean-Christophe OLIVO-MARIN[1], Alexandre DUFOUR*[1]

[1]Institut Pasteur, Bioimage Analysis Unit, CNRS UMR 3691, Paris, France

[2]Institut Jacques Monod, Membrane traffic in health & disease, CNRS UMR 7592, INSERM U950 Paris, France

daniel-felipe.gonzalez-obando@pasteur.fr, jcolivo@pasteur.fr, adufour@pasteur.fr

Résumé – Nous proposons une nouvelle implémentation parallèle des contours actifs pour la segmentation d'images combinant un système multi-agent avec une représentation du contour sous forme d'un "quad-edge". Les points de contrôle du contour évoluent indépendamment les uns des autres de manière parallèle, contrôlant ainsi la déformation du contour mais aussi sa convergence, tandis que la représentation "quad-edge" simplifie la manipulation du contour et sa re-paramétrisation locale lors de son évolution. Nous illustrons cette nouvelle approche sur les images biologiques et comparons les résultats avec une implémentation conventionnelle, en discutant des avantages et des limites actuelles. Ce travail préliminaire est disponible en tant que plug-in pour la plate-forme libre d'analyse d'images Icy.

Abstract – We investigate a novel, parallel implementation of active contours for image segmentation combining a multi-agent system with a quad-edge representation of the contour. The control points of the contour evolve independently from one another in a parallel fashion, handling contour deformation, and convergence, while the quad-edge representation simplifies contour manipulation and local re-sampling during its evolution. We illustrate this new approach on biological images, and compare results with a conventional implementation, discussing its benefits and limitations. This preliminary work is made available as a plug-in for the open-source Icy platform.

1 Introduction & Related work

Since their original appearance in computer vision 30 years ago [1], deformable models (also popularly referred to as *active contours*) have received extensive and continued attention from numerous scientific domains including bioimaging [2]. The curve is traditionally represented in one of two ways: a) explicitly, either via a parametric [1, 3] or a discrete [4, 5] formalism, or b) implicitly, by embedding the contour as the zero-level of a higher-dimensional Lipschitz function, a formalism well-known as level sets [6, 7]. The former approach is generally preferred for its interactivity and relative computational efficiency as compared to level sets (especially in 3D), while the latter approach is typically favoured for its topological flexibility and naturally multi-dimensional notation. These historical limitations have however been progressively addressed by the community, most notably with the introduction of topological constraints within the level set framework [8, 9], and conversely with the implementation of topological operations (splitting and merging) for discrete active contours [5, 10, 11].

Despite their flexibility and robustness, deformable models have long remained infamously known for their substantial computational burden. While the advent of GPU-oriented computing has enabled massively parallel implementations, notably for level sets [12–14], explicit approaches have only mildly benefited from GPU acceleration, with benefits mostly impacting heavy image-centric pre-processing operations, rather than the contour deformation itself [15–17]. To date, few alternatives to GPU implementations have been investigated. In [18], an original reformulation of the contour deformation was proposed using the concept of Multi-Agent-Systems [19], whereby all contour points behave pseudo-independently of one another. This concept is well suited for parallel computing, however it was not developed for computational efficiency, and therefore remained limited to a small number of agents (contour points). Also, convergence detection and local topological operations were not parallelized. More recently, a distributed approach was proposed in [20], where both the image and the contour are split into sub-images and sub-contours, thereby generating multiple sub-segmentation problems running in parallel. This solution is appealing for the analysis of very large data sets (typically surpassing both computer or graphics memory capacities), however the management of contour connectivity and fusion across neighboring sub-problems remains a challenge.

In this work we investigate for the first time a novel, parallel implementation of explicit active contours that draws from the theories of Multi-Agent Systems and the Quad-Edge formalism. The contributions of such a framework are two-fold:

- We propose an implementation of the contour deformation heavily inspired from Multi-Agent Systems (improving on the work of [18]), whereby in addition to handling their displacement and interaction with their neighbors, each control point is responsible for handling local re-

sampling operations (adding or removing control points) without the intervention of a global observer. We also improve on the convergence detection algorithm of each agent in order to significantly speed up the segmentation of complex objects.

- We represent the control points of the contour (i.e. the agents of the system) using the quad-edge formalism [21], which offers an efficient and elegant framework that simplifies contour manipulation and implementation. Moreover, the quad-edge formalism is readily adaptable to any dimension and contour topology, permitting the design of contour with complex geometries.

We describe in section 2 the general concept of our approach and its application to closed 2D contours, and report preliminary results in section 3. Benefits and limitations of the proposed approach are discussed in section 4, as well as its potential extensions and applications in biomedical imaging. Following reproducible research principles, the proposed algorithm is available in the form of a user-friendly plug-in in the open-source Icy bioimaging platform[1] [22].

2 Method

The starting point of our work is a fast, discrete implementation of multiple coupled self-resampling active contours with and without edges [5]. For the sake of simplicity, we illustrate the proposed approach using the 2D single-contour case without edges (other cases can be derived by analogy). We then present the two contributions of this work, namely our Multi-Agent strategy and the Quad-Edge implementation.

2.1 2D discrete active contours without edges

The general problem of object segmentation using active contours can be expressed as follows:

$$\arg\min_{\mathcal{C}} J(\mathcal{C}, I),\ s.t.\ J(\mathcal{C}, I) = J_{\text{data}}(\mathcal{C}, I) + J_{\text{reg}}(\mathcal{C}) \quad (1)$$

where $\mathcal{C}$ is the curve or contour evolving inside the image I, J_{data} is the data attachment term (which we derive here from the classical Chan-Vese-Mumford-Shah functional [23]), and J_{reg} is a regulariser of this ill-posed problem (here minimising local curvature [1]). In a discrete setting, the cost functional can be approximated by the sum of costs over the control points of the contour. Following a steepest gradient descent with explicit time-stepping, the iterative minimisation of J can be expressed as a set of forces applied to each control point of the contour $\mathcal{C}$ (see [5] for more details):

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \tau \cdot \left(\vec{\mathbf{f}}_{\text{data}}(\mathbf{x}_i^t, I) + \vec{\mathbf{f}}_{\text{reg}}(\mathbf{x}_i^t, \mathcal{C}_t) \right), \quad (2)$$

where t is the imaginary time discretisation variable representing the iterative minimisation process, τ is the minimisation time step, and $\mathbf{x}_i^t$ represents a control point of the contour $\mathcal{C}_t$ at iteration t. $\vec{\mathbf{f}}_{\text{data}}$ represents the data attachment term, reading

$$\vec{\mathbf{f}}_{\text{data}}(\mathbf{x}_i^t, I) = \left(|I(\mathbf{x}_i^t) - c_1(I, \mathcal{C}_t)|^2 - |I(\mathbf{x}_i^t) - c_2(I, \mathcal{C}_t)|^2 \right) \cdot \vec{\mathcal{N}}_i \quad (3)$$

where $I(\mathbf{x}_i^t)$ is the image value at the contour point (sampled with linear interpolation), c_1 and c_2 are the average intensities of I outside and inside $\mathcal{C}_t$, respectively, and $\vec{\mathcal{N}}_i$ is the outward-pointing unit normal vector to the contour at x_i^t. $\vec{\mathbf{f}}_{\text{reg}}$ represents the regularisation term, reading

$$\vec{\mathbf{f}}_{\text{reg}}(\mathbf{x}_i^t, \mathcal{C}_t) = \frac{\alpha}{2} \left(\mathbf{x}_{i-1}^t + \mathbf{x}_{i+1}^t - 2\mathbf{x}_i^t \right) \quad (4)$$

where $\mathbf{x}_{i-1}^t$ and $\mathbf{x}_{i+1}^t$) are the 2 neighbours of $\mathbf{x}_i^t$, and α is a non-negative weight balancing the influence between the data attachment and regularisation terms, chosen empirically.

We shall now describe below how this minimisation framework can benefit from a multi-agent implementation.

2.2 Multi-Agent Active Contours

Multi-Agent Systems (related to the field of distributed artificial intelligence [19]) are used to solve large, potentially intractable tasks using a set of cooperative agents that individually solve a sub-portion of the initial problem. It can be easily noticed that the problem described above is well suited to benefit from a Multi-Agent formalism, where the individual control points $\mathbf{x}_i$ can be seen as a swarm of individual agents evolving within the image space I under the influence of the forces defined in Eqs. 2, 3, and 4, until they minimise (as a whole) the target functional J. In practice, the Multi-Agent implementation is achieved by evolving each control point (or agent) in an independent thread handling local force computations and deformation (as suggested in [18]). However, 2 remaining tasks require a global synchronisation step and must be parallelised:

Convergence criterion: convergence is conventionally detected by globally monitoring the contour until contour displacement falls under a given $\epsilon > 0$. To parallelize this step, each control point now monitors its own stability during evolution, asynchronously. An additional benefit of this strategy is that control points can converge independently of one another, without necessarily waiting for a global criterion to be met. This heuristic criterion drastically reduces the computational load when the number of points is high, especially on complex objects, as we shall illustrate below.

Contour resampling Explicit active contours must be regularly re-parameterized throughout their evolution to ensure proper image sampling. This step is usually conducted every N iterations on the entire contour, by adding a new control point between neighboring points becoming too distant, or removing a control point that is too close to any of its neighbors [5,10,18]. We here again propose to defer this task to each control point, asynchronously. This however requires that the data structure holding the control points is well suited for this purpose, which is where the Quad-Edge formalism comes into play.

[1] http://icy.bioimageanalysis.org

2.3 Quad-Edge implementation

A quad-edge is a data structure used to model planar subdivisions. The elementary structure is an edge that stores its local topological and geometrical data [24]. In practice, the edge stores its end-points, the faces on each side, and holds a reference to its neighboring edges with same starting point (called the *O-ring*) and to one of its neighboring faces (called the *dual* of the edge). The structure is illustrated in Fig. 1. Quad-Edges maintain coherent references to points and faces on plane subdivisions, and the data structure permits efficient adjacency queries (neighboring edges are accessible in constant time) while local topological operations (point addition or deletion) is achievable in logarithmic time [21].

To summarize, starting from an initial contour, a global manager (the entry point of the algorithm) creates an edge for each control point, and ensures their connectivity. It then creates and assigns a separate thread for each control point, such that all points run in a fully autonomous manner, i.e. dealing with force computation, local re-sampling, and convergence detection. The global manager is responsible for updating global image-centric features (e.g. the average intensity inside and outside the contour, cf. Eq. 3), and for displaying the contour on screen. These two operations are also asynchronous, and are therefore run in the background on a regular basis.

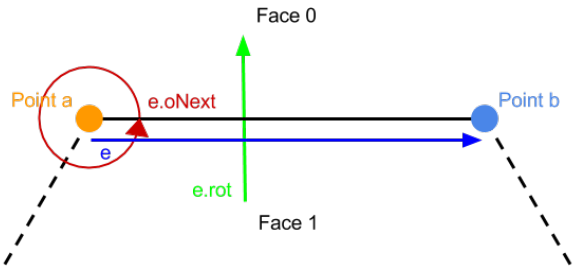

FIG. 1: Representation of a portion of a (closed) 2D contour using a Quad-Edge. Neighbouring edges are accessible in constant time (via *oNext*), while the "inside" of the contour is always known (via *rot*). This structure is used to facilitate the implementation of discrete active contours.

3 Experiments

We compared the performance of the proposed method against our non-parallel implementation [5]. Both algorithms are written in Java and available as ready-to-use plug-ins for the open source Icy platform [22]. Both algorithms are set to minimize the same cost functional (described in section 2) with the same parameters (initial contour, sampling, time step, weights). We performed all tests on a 2GHz quad-core processor, and report absolute times (best of 10 consecutive runs) as well as relative speed-up factor.

We first segment a simulated binary image of size 512×512 pixels containing a white disk of diameter 300 pixels in its center. Both algorithms start from a regular octagon of diameter 128 pixels in the image center. In this experiment, we impose a global convergence criterion on both algorithms (i.e. all control points evolve until the contour globally stabilizes), so as to measure solely the impact of parallel force computation and local contour re-sampling steps. Results are presented in Table 1. It can be seen that for few control points, the parallel implementation is slower due to the overhead of creating individual threads for parallel processing, which is non negligible in comparison to the total computation time. This trend quickly reverses as soon as the number of control points increases, with the parallel implementation yielding up to more than double the performance of its non-parallel counterpart.

Sampling (px)	16	8	4	2	1
Nb. control points	54	102	197	383	783
Baseline [5] (ms)	97	123	215	631	2251
Proposed (ms)	121	117	179	336	952
Gain factor	0.8×	1.0×	1.2×	1.9×	2.4×

TAB. 1: Performance comparison between parallel and non-parallel active contour implementations, with global convergence detection. The number of points is given after convergence and is the same for both algorithms.

In a second example, we now segment a real biological image of a neuron (cf. Fig. 2). Here we compare two variants of the proposed algorithm to the baseline: one with global convergence detection (similar to the previous example, which we refer to as "semi-parallel"), and the fully parallel implementation, where control points also handle convergence asynchronously. Results are presented in Table 2. While the semi-parallel version is not noticeably faster, the fully parallel implementation clearly outperforms the baseline method, which is well illustrated on such complex biological objects, where the vast majority of control points converge rapidly (around the soma and along the branches), while only a handful of control points remain "active" (at the leading edge).

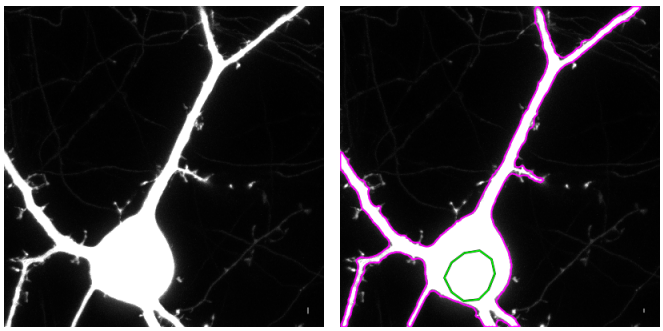

FIG. 2: Fluorescence microscopy image of a neurone in culture (image size: 400×400). Left: original image. Right: initial contour (green) and segmentation result (magenta) superimposed on the original image. Contour sampling: 2 pixels.

Sampling (px)	4	2	1
Nb. control points	344	651	1253
Baseline [5] (ms)	497	1443	6844
Semi-parallel (ms)	827 (0.6×)	2067 (0.7×)	6536 (1.1×)
Fully-parallel (ms)	715 (0.7×)	1010 (1.4×)	2086 (3.3×)

TAB. 2: Performance comparison of two version of the proposed Quad-Edge Parallel active contours against the equivalent, non-parallel implementation on a real biological image (cf. Fig. 2). Convergence detection is global in the semi-parallel case, and local in the fully parallel case.

4 Discussion

We have presented a novel, parallel implementation of discrete active contours using the concept of Multi-Agent Systems and the Quad-Edge formalism, whereby the evolution of the control points is fully asynchronous, from force computation to local topological re-sampling and convergence detection. The segmentation results are comparable, while the computational performance is substantially improved, yet without the need for a GPU-specific implementation. We expect that these preliminary but promising results will enable new and exciting developments in the field of active contours. Indeed, in addition to being GPU-friendly, the Quad-Edge implementation offers great topological flexibility. This facilitates the extension of the method to 3D meshes and open contours, which we shall investigate in subsequent work.

Acknowledgements

This work was funded by Institut Pasteur. L.R. was partially funded by an interdisciplinary grant from the CNRS GdR MIV.

References

[1] Michael Kass, Andrew Witkin, and Demetri Terzopoulos, "Snakes: Active contour models," 1988.

[2] Christophe Zimmer, Bo Zhang, Alexandre Dufour, Aymeric Thebaud, Sylvain Berlemont, Vannary Meas-Yedid, and Jean-Christophe Olivo-Marin, "On the digital trail of mobile cells," *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 54–62, 5 2006.

[3] Christophe Zimmer and Senior Member, "Coupled Parametric Active Contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1838–1842, 2005.

[4] Andrei C Jalba, Michael H F Wilkinson, and Jos B T M Roerdink, "CPM : A Deformable Model for Shape Recovery and Segmentation Based on Charged Particles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1320–35, 2004.

[5] Alexandre Dufour, Roman Thibeaux, Elisabeth Labruyère, Nancy Guillén, and Jean-Christophe Olivo-Marin, "3D active meshes: fast discrete deformable models for cell tracking in 3D time-lapse microscopy.," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 1925–37, 7 2011.

[6] James A. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, 2nd editio edition, 1999.

[7] Alexandre Dufour, Vasily Shinin, Sharagim Tajbaksh, Nancy Guillén, Jean-Christophe Olivo-Marin, and Christophe Zimmer, "Segmenting and tracking fluorescent cells in dynamic 3D microscopy with coupled active surfaces," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1396–1410, 2005.

[8] Xiao Han, Student Member, Chenyang Xu, Jerry L Prince, and Senior Member, "A Topology Preserving Level Set Method for Geometric Deformable Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 755–768, 2003.

[9] Florent Ségonne, "Active Contours Under Topology Control — Genus Preserving," *International Journal of Computer Vision*, vol. 79, no. 2, pp. 107–17, 2008.

[10] Jacques-olivier Lachaud and Annick Montanvert, "Deformable meshes with automated topology changes for coarse-to-fine three-dimensional surface extraction," *Medical Image Analysis*, vol. 3, no. 2, pp. 187–207, 1999.

[11] Tim Mcinerney and Demetri Terzopoulos, "T-snakes : Topology adaptive snakes," *Medical Image Analysis*, vol. 4, pp. 73–91, 2000.

[12] Joshua E Cates, Aaron E Lefohn, and Ross T Whitaker, "GIST: an interactive , GPU-based level set segmentation tool for 3D medical images," *Medical Image Analysis*, vol. 8, pp. 217–231, 2004.

[13] Mike Roberts, Jeff Packer, Mario Costa Sousa, and Joseph Ross Mitchell, "A Work-Efficient GPU Algorithm for Level Set Segmentation," in *Proceedings of High Performance Graphics*, 2010.

[14] Julian Lamas-Rodriguez, Dora B. Heras, Francisco Arguello, Dagmar Kainmueller, Stefan Zachow, and Montserrat Boo, "GPU-accelerated level-set segmentation," *Journal of Real-Time Image Processing*, vol. 12, pp. 15–29, 2016.

[15] O C Eidheim, J Skjermo, and L Aurdal, "Real-time analysis of ultrasound images using GPU," *International Congress Series*, vol. 1281, pp. 284–289, 2005.

[16] Jérôme Schmid, José A Iglesias, and Enrico Gobbetti Nadia Magnenat-thalmann, "A GPU framework for parallel segmentation of volumetric images using discrete deformable models," *The Visual Computer*, vol. 27, pp. 85–95, 2011.

[17] Rigo Alvarado, Juan J Tapia, and Julio C Rolon, "Medical image segmentation with deformable models on graphics processing units," *The Journal of Supercomputing*, vol. 68, no. 1, pp. 339–64, 2014.

[18] Abdelkader Fekir and Nacéra Benamrane, "Segmentation of Medical Image Sequence by Parallel Active Contour," in *Software Tools and Algorithms for Biological Systems*, pp. 515–522. Springer, 2011.

[19] Nikos Vlassis, *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*, Morgan & Claypool, 1st editio edition, 2007.

[20] Ping Jiang, Quansheng Dou, and Xiaoying Hu, "A Parallel Realization of the Active Contour Model on Boundary Extraction," *Applied Mathematics & Information Science*, vol. 260, no. 1, pp. 253–260, 2014.

[21] A Gouaillard, L Florez-Valencia, and E Boix, "itkQuadEdgeMesh : A Discrete Orientable 2-Manifold Data Structure for Image Processing," *The Insight Journal*, pp. 1–19, 2006.

[22] Fabrice de Chaumont, Stéphane Dallongeville, Nicolas Chenouard, Nicolas Hervé, Sorin Pop, Thomas Provoost, Vannary Meas-Yedid, Praveen Pankajakshan, Timothée Lecomte, Yoann Le Montagner, Thibault Lagache, Alexandre Dufour, and Jean-Christophe Olivo-Marin, "Icy: an open bioimage informatics platform for extended reproducible research.," *Nature Methods*, vol. 9, no. 7, pp. 690–6, 2012.

[23] Tony F Chan and Luminata A Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, pp. 266–277, 2001.

[24] Leonidas Guibas and Jorge Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams," *ACM Transactions on Graphics*, vol. 4, no. 2, pp. 74–123, 1985.