Quad-edge active contours for biomedical image segmentation

Daniel Felipe Gonzalez Obando, Lauriane C Rohfritsch, Manon C Faure, Lydia C Danglot, Vannary C Meas-Yedid, Jean-Christophe C Olivo-Marin, Alexandre C Dufour

QUAD-EDGE ACTIVE CONTOURS FOR BIOMEDICAL IMAGE SEGMENTATION

Daniel F Gonzalez[1], *Lauriane Rohfritsch*[1,2], *Manon Faure*[1], *Lydia Danglot*[2]
Vannary Meas-Yedid[1], *Jean-Christophe Olivo-Marin*[1] *and Alexandre C Dufour*[1]

[1]Institut Pasteur, Bioimage Analysis Unit, CNRS 3691, Paris, France
[2]Institut Jacques Monod, Membrane traffic in health & disease, CNRS 7592, Inserm 950, Paris, France

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 active contour implementation, discussing its benefits and limitations. This preliminary work is made freely available as a plug-in for our open-source Icy platform, where it will be developed with future extensions.

***Index Terms*—** Image segmentation, active contours, quad edge, parallel computing, multi-agent systems

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 biomedical imaging, with applications ranging from image segmentation to object tracking, shape modelling and matching [2–4]. Briefly, the principle is to deform an initial curve towards a target object boundary by minimising some cost functional that comprises data-attachment and regularisation terms, as well as user-defined priors to ensure optimal segmentation. The curve is traditionally represented in one of two ways: a) explicitly, either via a parametric [1, 5, 6] or a discrete [7–9] 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 [10–13]. 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 [14–16], and conversely with the implementation of topological operations (splitting and merging) for discrete active contours [9, 17, 18].

Correspondence: adufour@pasteur.fr

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

Despite their flexibility and robustness, deformable models have long remained infamously known for their substantial computational burden as compared to simpler yet faster alternatives. Fortunately, this situation has drastically evolved over the last decade, notably with the advent of GPU[1]-oriented computing [19]. Level sets in particular have largely benefited from such massively parallel implementations [20–22]. Explicit approaches have also benefited from GPU acceleration, although to a lesser extent, with benefits mostly impacting heavy image-centric pre-processing operations, rather than the contour deformation itself [23–25]. Aside from such GPU-centric approaches, very few alternatives have been investigated. In [26], an original reformulation of the contour deformation was proposed using the concept of Multi-Agent-Systems [27], whereby all contour points behave pseudo-independently of one another. This concept carries high potential for parallel computing (independently of the hardware), however the approach 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 are not parallelised. More recently, a distributed approach was proposed in [28], 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 particularly 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 accross neighbouring 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 [26]), whereby in addition to handling their displacement and interaction with their neighbours, each control point is responsible for handling local resampling 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 [29, 30], which offers an efficient and elegant framework that simplifies con-

[1]Graphical Processing Unit

tour manipulation and implementation (notably in a parallel context). 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, including a quantitative comparison of the proposed approach with the equivalent, non-parallel formalism. We finally discuss the benefits and limitations of the proposed approach 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 our open-source Icy bioimaging platform[2] [31].

2. METHOD

The starting point of our work is a fast, discrete implementation of multiple coupled self-resampling active contours with and without edges [9]. For the sake of simplicity, yet without loss of generality, 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 [32]), 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 [9] 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), while c_1 and c_2 are the average intensities of I outside and inside $\mathcal{C}_t$, respectively. $\vec{\mathbf{f}}_{\text{reg}}$ 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.

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

[2] http://www.bioimageanalysis.org

2.2. Multi-Agent Active Contours

The principle of Multi-Agent Systems (a field of research closely related to distributed artificial intelligence [27]) is to carry out a large, potentially intractable task, 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 [26]). However, 2 remaining tasks require a global synchronisation step and must be parallelised:

- **Convergence criterion** In our previous implementation, we detect convergence by globally monitoring the contour in a periodic manner (typically after every iteration), assuming that convergence is reached whenever the change in contour area from the previous iteration falls under a given $\epsilon > 0$. To parallelise this step, we first replace this criterion by monitoring instead the stability of each control point over time, which can be done 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, and is most useful when segmenting complex objects, as we shall illustrate below.
- **Contour resampling** Explicit active contours need to be regularly resampled (or re-parameterised) throughout their evolution to ensure proper image sampling. In previous works, the contour was globally resampled at the end of every iteration, by adding a new control point between neighbouring points becoming too distant, or removing a control point that is too close to any of its neighbours [9, 17, 26]. We propose here to confer this resampling ability to the control points themselves, which already have a dependency on their immediate neighbours for force computation. This asynchronous step 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.

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 [29]. In practice, the edge stores its end-points, the faces on each side, and holds a reference to its neighbouring edges with same starting point (called the *O-ring*) and to one of its neighbouring faces (called the *dual* of the edge). The structure is illustrated in Fig. 1. From a geometrical point of view, quad-edges allow keeping coherent references to points and faces on plane subdivisions. Moreover, this data structure permits efficient adjacency queries (neighbouring edges are accessible in constant time) and local topological operations (point addition or deletion is achievable in logarithmic time [30]).

To summarise, starting from an initial contour (e.g. a region of interest), 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 resampling, and convergence detection. The global manager is responsible for updating global image-centric features (notably the average intensity inside and outside the contour, cf. Eq. 3), and for displaying the contour on screen. These two operations do not require a synchronisation step, and are therefore run in the background on a regular basis.

3. EXPERIMENTS

We illustrate here the performance of the proposed implementation, in comparison to our previous approach [9]. Both algorithms have been in written in Java and are available as ready-to-use plug-ins for the open source Icy platform [31]. Both algorithms were set to minimise the same cost functional (described in section 2), and were given the same parameters (initial contour, sampling, time step, weights). We performed all tests on a 2GHz quad-core processor, and report both absolute times (best of 10 consecutive runs) as well as relative times between implementations for more clarity, taking our previous implementation as the baseline.

We start by segmenting a simulated binary image of size 512×512 pixels containing a circle of diameter 300 pixels in its centre. We initialise both algorithms with a regular octagon place in the centre of the circle with a diameter of 128 pixels. In this experiment, we imposed a global convergence criterion on both algorithms (i.e. all control points evolve until the contour globally stabilises), so as to measure solely the impact of parallelising the force computation and local contour resampling steps. Results are presented in Table 1. It can be seen that for contours with few control points, the parallel implementation is slightly slower, this is 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 sampling rate decreases to a more realistic value (thus increasing the total number of points), where the parallel implementation can yield more than double the performance of its non-parallel counterpart.

In a second example, we now segment a real biological image representing a neurone 2. In this example, we compare two variants of the proposed algorithm to the baseline: one with global convergence detection (similar to the previous example), and the fully parallel implementation, where control points handle convergence asynchronously. Results are presented in Table 2. Given the object complexity, computation times are higher than in the previous example. While the semi-parallel version is not noticeably faster, the fully parallel implementation clearly outperforms the baseline method. The gain factor is well illustrated on such complex biological objects, since the vast majority of the control points converge rapidly (around the soma and along the branches), while most of the "active" computation occurs solely at the leading edge of the contour, thus concerning only a handful of control points.

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 resampling and convergence detection. By reworking the core components of the contour optimisation process, we were able to significantly speed up the segmentation process, yielding improved performance compared to the standard implementation even without the need for

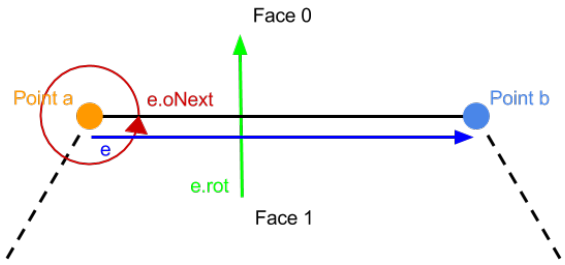

Fig. 1. Representation of a portion of a (closed) 2D contour using a Quad-Edge. Neighbouring edges are accessible in constant time (via *oNext*), while the "inside" of the contour is always known (via *rot*). This structure is used to facilitate the implementation of discrete active contours.

Sampling (px)	16	8	4	2	1
Nb. control points	54	102	197	383	783
Baseline [9] (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×

Table 1. Performance comparison between the proposed active contour implementation and the equivalent, non-parallel algorithm. The number of points is given after convergence and is the same for both algorithms.

Sampling (px)	4	2	1
Nb. control points	344	651	1253
Baseline [9] (ms)	497	1443	6844
Proposed, GC (ms)	827 (0.6×)	2067 (0.7×)	6536 (1.1×)
Proposed, LC (ms)	715 (0.7×)	1010 (1.4×)	2086 (3.3×)

Table 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). GC: semi parallel version with global convergence detection. LC: fully parallel version with local convergence detection.

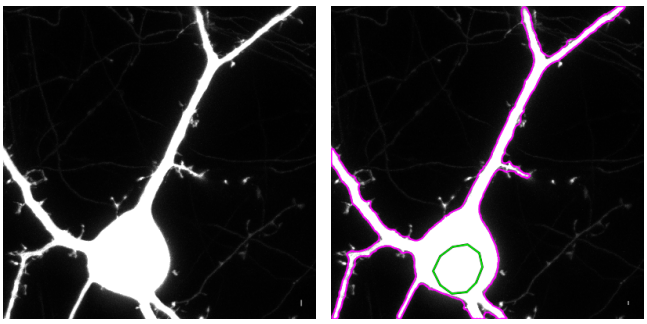

Fig. 2. Fluorescence microscopy image of a neurone in culture (image size: 400×400). Left: original image. Right: initial contour (green) and segmentation result (magenta) superimposed on the original image. Contour sampling: 2 pixels.

a GPU-specific implementation. Thanks to the Quad-Edge formalism, we expect that these preliminary but promising results will lead us to a number of 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. Not only does this facilitate the extension of the method to 3D meshes, but the Quad-Edge structure can be adapted to handle open contours as well as contours with a hybrid topology (e.g. a combination of closed, open, 2D and 3D geometries). Our future work will investigate these extensions and their applications in biomedical imaging.

5. REFERENCES

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

[2] Tim Mcinerney and Demetri Terzopoulos, "Deformable models in medical image analysis : a survey," *Medical Image Analysis*, vol. 1, no. 2, pp. 91–108, 1996.

[3] Christophe Zimmer, Elisabeth Labruyère, Vannary Meas-Yedid, Nancy Guillén, and Jean-Christophe Olivo-Marin, "Segmentation and tracking of migrating cells in videomicroscopy with parametric active contours: a tool for cell-based drug testing," *IEEE Transactions on Medical Imaging*, vol. 21, no. 10, pp. 1212–1221, oct 2002.

[4] Alexandre Dufour, Tzu-yu Liu, Christel Ducroz, Robin Tournemenne, Beryl Cummings, Roman Thibeaux, Nancy Guillen, Alfred O Hero, and Jean-Christophe Olivo-Marin, "Signal Processing Challenges in Quantitative 3-D Cell Morphology," *IEEE Signal Processing Magazine*, vol. 1, no. January, pp. 30–40, 2015.

[5] 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.

[6] Ricard Delgado-gonzalo, Virginie Uhlmann, Daniel Schmitter, and Michael Unser, "Snakes on a Plane: a perfect snap snap for bioimage analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 1, pp. 41–48, 2015.

[7] 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.

[8] Xianghua Xie and Majid Mirmehdi, "MAC : Magnetostatic Active Contour Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 632–647, 2008.

[9] 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, jul 2011.

[10] Viccnt Caselles, Francine Catt, Tomeu Coll, and Françoise Dibos, "A geometric model for active contours in image processing," *Numerische Matematik*, vol. 31, pp. 1–31, 1993.

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

[12] 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.

[13] Nikos Paragios, "Curve Propagation, Level Set Methods and Grouping," in *Handbook of Biomedical Imaging*, pp. 45–62. Springer, 2015.

[14] 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.

[15] Chih-yu Hsu, Chih-hung Yang, and Hui-ching Wang, "Topological control of level set method depending on topology constraints," *Pattern Recognition Letters*, vol. 29, pp. 537–46, 2008.

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

[17] 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.

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

[19] Erik Smistad, Thomas L Falch, Mohammadmehdi Bozorgi, Anne C Elster, and Frank Lindseth, "Medical image segmentation on GPUs A comprehensive review," *Medical Image Analysis*, vol. 20, no. 1, pp. 1–18, 2015.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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