Multi-Object Geodesic Active Contours (MOGAC): A Parallel Sparse-Field Algorithm for Image Segmentation

Blake C. Lucas
Johns Hopkins University
Baltimore, MD
blake@cs.jhu.edu

Michael Kazhdan
Johns Hopkins University
Baltimore, MD
misha@cs.jhu.edu

Russell H. Taylor
Johns Hopkins University
Baltimore, MD
rht@jhu.edu

Abstract

An important task for computer vision systems is to segment adjacent structures in images without producing gaps or overlaps. Multi-object Level Set Methods (MLSM) perform this task with the benefit of sub-pixel accuracy. However, current implementations of MLSM are not as computationally or memory efficient as their region growing and graph cut counterparts which lack sub-pixel accuracy. To address this performance gap, we present a novel parallel implementation of MLSM that leverages the sparse properties of the segmentation algorithm to minimize its memory footprint for multiple objects. The new method, Multi-Object Geodesic Active Contours (MOGAC), can represent N objects with just two functions: a label image and unsigned distance field. The time complexity of the algorithm is shown to be O((M^d)/P) for M^d pixels and P processing units in dimension d={2,3}, independent of the number of objects. Results are presented for 2D and 3D image segmentation problems.

1. Introduction

The Level Set Method (LSM) [1, 2] is popular in computer vision systems for segmenting images [3]. LSM solves PDEs to produce image segmentations with sub-pixel accuracy. The multi-object version is capable of segmenting adjacent structures without gaps or overlaps [4-9]. However, current implementations of the Multi-object Level Set Method (MLSM) are slow and require a large memory footprint compared to their region growing [10] and graph cut counterparts [11], which lack sub-pixel accuracy. A modern challenge is to develop MLSM implementations that have competitive computational and memory efficiency with region growing and graph cut methods.

2. Related Work

Several methods have been proposed for segmenting N objects with N level set functions [4, 6, 8] that are stored as images. Storage of these level set images is intractable for tasks such as cell tracking in microscopy images [12] where there are potentially hundreds to thousands of objects. The Multi-phase LSM [7] reduces the number of level sets to $log_2(N)$, and the Multi-compartment LSM [5, 9] reduces the number of functions to 4 in 2D and 6 in 3D. Even with these advancements, some segmentation tasks are still intractable because the time complexity for existing MLSMs is dependent on the number of objects. By comparison, region growing techniques require only one function to represent N objects, and the computation time can be independent of the number of objects.

All previous MLSMs use serial implementations of the narrow-band method [1], which requires periodic re-initialization of the signed distance field. One notable exception is the work by Lie et al. [13], but their method does not have sub-pixel precision. The fast-marching method [14] for distance field re-initialization is a computational bottleneck whose complexity is $O(M^d logM)$ for M^d pixels in dimension $d \in \{2,3\}$. A more efficient approach is to use the sparse-field method [15] that has $O(M^{d-1})$ time complexity. The sparse-field LSM stores only the minimum narrow-band needed for finite difference calculations and maintains an approximation to the signed distance field at every time step. The sparse-field LSM is competitive with region growing methods and will be extended in this work to create a new MLSM.

To develop LSMs that run at faster, real-time speeds, implementations must leverage parallelism now abundant on modern GPUs and CPUs. There has been work on parallel implementations of single object LSMs [16-18] that achieve substantial speed-up. Memory consumption is a concern for these parallel implementations because GPUs generally have access to less memory than CPUs; and at real-time speeds, memory latency and access patterns become a major performance concern. These concerns complicate development of a parallel MLSM, which has yet to be proposed in literature. We regard parallelism as a necessary consideration when proposing a new algorithm because computing hardware is becoming more parallel as opposed to becoming faster.

3. Overview

The contributions of this work are two-fold. First, we describe how to represent N level sets with only two functions. Second, we present a parallel implementation of the sparse-field LSM for segmenting N objects in 2D and 3D. Properties of the sparse-field LSM are leveraged to represent and evolve N level sets with only two functions.

Results are presented for 2D/3D segmentation of multiple, potentially overlapping, objects. The scalability of the algorithm is analyzed, and the computational complexity of the algorithm is discussed and juxtaposed with other MLSM implementations. The algorithm is implemented as a mixture of Java and OpenCL, and the source code is distributed as part of the Java Image Science Toolkit[1] [19] to encourage development of new image analysis tools.

4. Method

4.1. Representation

The following LSM, which we refer to as Multi-Object Geodesic Active Contours (MOGAC), segments a gray level image $I: \Omega \longmapsto \mathcal{R}$ with domain $\Omega \subset \mathcal{R}^d$ in dimension $d \in \{2,3\}$ into at most N object regions represented by signed distance fields $\varphi_n: \Omega \longmapsto \mathcal{R}$ for labeled regions $n \epsilon \mathcal{L} = \{1, \cdots, N\}$. The segmentation is compressed into a label function $\chi: \Omega \longmapsto \mathcal{L}$ and unsigned distance field $\psi: \Omega \longmapsto \mathcal{R}$:

$$\psi(\boldsymbol{x}) = \min_n |\varphi_n(\boldsymbol{x})|. \qquad (1)$$

Fig. 1 depicts both the unsigned distance field and label image representing 5 objects that overlap to create a total of $N = 8$ different object regions.

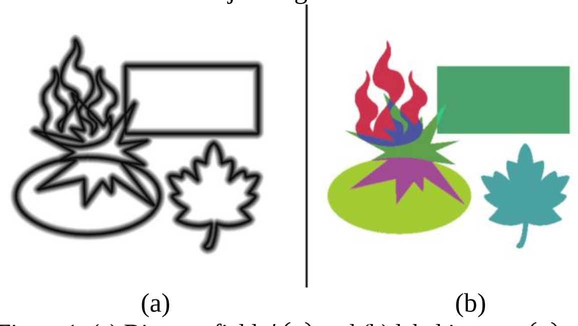

(a) (b)

Figure 1: (a) Distance field $\psi(\boldsymbol{x})$ and (b) label image $\chi(\boldsymbol{x})$

An approximation to $\varphi_n(\boldsymbol{x})$ is computable from $\chi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$ at the boundary of region n:

$$\Lambda_n = \{\boldsymbol{x} |\ \exists \boldsymbol{y} \epsilon \mathcal{N}(\boldsymbol{x}) \text{ s.t. } \sigma_n(\boldsymbol{x}) \neq \sigma_n(\boldsymbol{y})\} \qquad (2)$$

where $\mathcal{N}(\boldsymbol{x})$ is the $2d$-connected neighborhood of pixel $\boldsymbol{x}$ and the sign $\sigma_n(\boldsymbol{x})$ is indicated by,

$$\sigma_n(\boldsymbol{x}) = \begin{cases} -1 & \chi(\boldsymbol{x}) = n \\ 1 & otherwise \end{cases}. \qquad (3)$$

The partially reconstructed level set $\tilde{\varphi}_n : \Lambda_n \longmapsto \mathcal{R}$ is given by,

$$\tilde{\varphi}_n(\boldsymbol{x}) = \sigma_n(\boldsymbol{x})\, \psi(\boldsymbol{x}). \qquad (4)$$

Eq. 4 accurately measures the signed distance at the boundary of two objects. At the shared boundary of more than two objects, this measurement is approximate. $\tilde{\varphi}_n$ provides enough information to extract an iso-surface with marching cubes [20] or recover the entire signed distance field with fast-marching [14]. As previously mentioned, we want to avoid fast-marching to save time and memory. Therefore, the level set evolution scheme must restrict its computational domain to Λ_n for each level set φ_n. The sparse-field LSM [15] has exactly this property.

4.2. Level Set Evolution

The N objects represented by N level sets $\vec{\varphi} = \{\varphi_1(\boldsymbol{x}), \cdots, \varphi_N(\boldsymbol{x})\}$ are evolved through time by solving the following differential equation:

$$\frac{\partial \vec{\varphi}(x,t)}{\partial t} = \vec{f}(\boldsymbol{x}, t)\boldsymbol{\delta}\big(\vec{\varphi}(\boldsymbol{x}, t)\big), \qquad (5)$$

where $\vec{f}(\boldsymbol{x}, t) \in \mathcal{R}^N$ is the speed function and $\boldsymbol{\delta}(\cdot)$ is an $N \times N$ diagonal matrix whose diagonal entries are compactly supported approximations to the dirac delta $\delta(\cdot)$. The i^{th} diagonal entry $\boldsymbol{\delta}_{ii}(\vec{\varphi}) = \delta(\varphi_i)$. Subsequent examples use speed functions of the following form:

$$f_n(\boldsymbol{x}) = \lambda_\rho \rho(\boldsymbol{x}) \|\nabla\varphi_n(\boldsymbol{x})\| + \lambda_v \vec{v}(\boldsymbol{x}) \cdot \nabla\varphi_n(\boldsymbol{x}) + \lambda_\kappa \kappa_n(\boldsymbol{x}) \|\nabla\varphi_n(\boldsymbol{x})\|, \qquad (6)$$

where $\rho(\boldsymbol{x}) \in \mathcal{R}$ is a pressure force that drives the object's boundary towards a particular image intensity, $\vec{v}(\boldsymbol{x}) \in \mathcal{R}^d$ is an external velocity field that drives the boundary towards edges in the image, and $\kappa_n(\boldsymbol{x}) \in \mathcal{R}$ is the mean curvature for object n. Relative contributions of each force are controlled by weights λ_ρ, λ_v, and λ_κ. Forces are computed with first-order upwind finite differences [1] on either a 3×3 or $3 \times 3 \times 3$ stencil in 2D or 3D respectively. A first-order solution to eq. 5 yields the following iterative scheme:

$$\vec{\varphi}(\boldsymbol{x}, t + \Delta t) = \vec{\varphi}(\boldsymbol{x}, t) + \Delta t \vec{f}(\boldsymbol{x}, t)\boldsymbol{\delta}\big(\vec{\varphi}(\boldsymbol{x}, t)\big). \qquad (7)$$

The evolution process changes the location of the zero iso-level for each object, thereby moving object boundaries in accordance with the speed function $\vec{f}$. After computing updates for all level sets $\vec{\varphi}(\boldsymbol{x})$, the result must be stored in the label and unsigned distance images $\chi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$ respectively. To do this, we use the projection method proposed by Losasso et al. [8]. New level set values are sorted to find the smallest two $\varphi_a(\boldsymbol{x})$ and $\varphi_b(\boldsymbol{x})$ s.t.

[1] Please visit http://www.nitrc.org/projects/jist

$\varphi_a(\boldsymbol{x}) < \varphi_b(\boldsymbol{x})$ among those labels n for which $\boldsymbol{x} \epsilon \Lambda_n$. The label and unsigned distance images are updated via:

$$\chi(\boldsymbol{x},t) = \begin{cases} b & \varphi_a(\boldsymbol{x},t) = \varphi_b(\boldsymbol{x},t) \text{ and } b < a \\ a & otherwise \end{cases} \tag{8}$$

and

$$\psi(\boldsymbol{x},t) = \frac{1}{2}|\varphi_a(\boldsymbol{x},t) - \varphi_b(\boldsymbol{x},t)|. \tag{9}$$

This projection technique has several useful properties. First, it reduces the complexity of topological relationships between N objects in any finite dimension to just two objects in one dimension. Second, it insures objects within a distance of 1 pixel from each other do not overlap or have air gaps. Third, it couples forces between adjacent objects so that the shared boundary can move without creating gaps or overlaps.

The initial segmentation may have overlaps even though the final image segmentation should not. One goal of level set evolution is to remove these overlaps. For this task, overlapping object regions in the initial segmentation are treated as different objects using bit masks. If a pixel belongs to object k, then there will be a 1 in the k^{th} bit position of label n. A label image stored as 32 bit integers can distinguish between 32 distinct objects and 2^{32} different combinations of objects. It is then necessary to define forces that contract overlapping object regions so that the final segmentation contains only distinct objects.

To evolve $\tilde{\varphi}_n(\boldsymbol{x})$, we evaluate eq. 7 on the subset Γ_n (eq. 10), by windowing $\delta(\cdot)$ to have support $[-0.5,0.5]$ pixels.

$$\Gamma_n = \{\boldsymbol{x}| -0.5 \leq \tilde{\varphi}_n(\boldsymbol{x}) \leq 0.5, \forall \boldsymbol{x} \epsilon \Omega\} \subseteq \Lambda_n \tag{10}$$

The CFL condition [21] is enforced by choosing Δt s.t. $\Delta t \leq 0.5/f_{max}$ where $f_{max} = \max_{n,x}\left|f_n(\boldsymbol{x})\delta\big(\psi(\boldsymbol{x},t)\big)\right|$. To compute $\vec{f}(\boldsymbol{x},t)$ with finite differences, $\varphi_n(\boldsymbol{x})$ must be known in the neighborhood around $\boldsymbol{x}$. If $\boldsymbol{x} \in \Lambda_n$ and $\chi(\boldsymbol{x}) = n$, it is true that $\forall \boldsymbol{y} \in \mathcal{N}(\boldsymbol{x})$, $\varphi_n(\boldsymbol{y}) = \sigma_n(\boldsymbol{y})\, \psi(\boldsymbol{y})$ because either $\chi(\boldsymbol{y}) \neq \chi(\boldsymbol{x})$ which implies $\boldsymbol{y} \in \Lambda_n$, or $\chi(\boldsymbol{y}) = \chi(\boldsymbol{x})$ which implies $\boldsymbol{y}$ is inside object n, so $\psi(\boldsymbol{y})$ must be a measurement to object n. If $\boldsymbol{x} \in \Lambda_n$ but $\chi(\boldsymbol{x}) = m$ and $\chi(\boldsymbol{y}) = l$ s.t. $n, m,$ and l are all different, $\psi(\boldsymbol{y})$ could be a measurement to object l instead of n. The problem can be resolved by setting $\varphi_n(\boldsymbol{y}) = (\psi(\boldsymbol{x}) + 1)\sigma_n(\boldsymbol{y})$ if $\boldsymbol{y} \notin \Lambda_n$. To address the problem in general for larger neighborhoods, we recommend the Multi-compartment LSM [5, 9]. A solution is not implemented in this work because the problem is unnoticeable in practical examples. Future work will implement a solution.

The level set evolution process is described by *Algorithm 1*. The unsigned distance field ψ is rebuilt (**Rebuild**) within a distance of $L = 3$ pixels via the fast approximation described in [15] (see *Algorithm 2)*. **Evolve** is straightforward to parallelize because each for-loop over Ω is dependent on only the $2d$-connected neighborhood around each pixel. The only step that is non-trivial to parallelize is computation of f_{max}, which can be done with a parallel reduction [22] in $O(M^d log_2(M)/P)$ time. This computation can be avoided by crafting $\vec{f}$ s.t. $f_n(\boldsymbol{x},t) \in [-1,1]$ or clamping $\tau_k(\boldsymbol{x})$ to the range $[-1,1]$. Clamping is an acceptable shortcut in image segmentation problems because forces do not have to be physically accurate. Furthermore, forces only need to be evaluated for objects that compete for a particular pixel, which is at most $(2d + 1)$. The computational complexity of **Evolve** is then $O(M^d/P)$ for M^d pixels and P processing units. Details of the implementation are contained in the open-source release, which does compute f_{max}.

Algorithm 1. **Evolve**

```
  foreach $\boldsymbol{x} \in \Omega$ do
    if $\exists n$ s.t. $\boldsymbol{x} \in \Gamma_n$ then
// Compute speeds for all pixels in the active set for every
// pair of adjacent objects.
      foreach $\boldsymbol{y}_k \epsilon (\mathcal{N}(\boldsymbol{x}) \cup \{\boldsymbol{x}\})$ do
        $n = \chi(\boldsymbol{y}_k, t)$
        $\tau_k(\boldsymbol{x}) = f_n(\boldsymbol{x},t)\delta\big(\varphi_n(\boldsymbol{x},t)\big)$
        $l_k(\boldsymbol{x}) = n$
// Find the maximum speed.
  $f_{max} = \max_{k,x}|\tau_k(\boldsymbol{x})|$
  if $f_{max} > 1$ then $\Delta t = 0.5/f_{max}$ else $\Delta t = 0.5$
  foreach $\boldsymbol{x} \in \Omega$ do
    if $\exists n$ s.t. $\boldsymbol{x} \in \Gamma_n$ then
      $Z = \emptyset$
// Compute level set updates for objects that compete for pixel $\boldsymbol{x}$.
      for $k = 1{:}(2d+1)$ do
        $n = l_k(\boldsymbol{x})$
        $z = \tilde{\varphi}_n(\boldsymbol{x},t) + \Delta t \tau_k(\boldsymbol{x})$
        if $(z,n) \notin Z$ then $Z = Z \cup \{(z,n)\}$
// Resolve air-gaps and overlaps with projection technique.
      Sort $Z$ by $z$ to find $\varphi_a(\boldsymbol{x})$ and $\varphi_b(\boldsymbol{x})$
// Store the new level set value.
      Compute $\chi(\boldsymbol{x}, t+\Delta t)$ and $\psi(\boldsymbol{x}, t+\Delta t)$
  Rebuild
```

Algorithm 2. **Rebuild**

```
foreach $l = 1{:}L$ do
  foreach $\boldsymbol{x} \in \Omega$ do
// if $\boldsymbol{x}$ is not in the previous active set and the new distance
// measurement has not been computed yet.
    if $\psi(\boldsymbol{x},t) > 0.5$
      and $\psi(\boldsymbol{x},t+\Delta t) > (l - 0.5)$ then
// Estimate the minimum distance to the closest object.
      $\psi(\boldsymbol{x},t+\Delta t) =$
        $\min_{y \epsilon \mathcal{N}(x)}\left|\sigma_{\chi(x)}(\boldsymbol{y},t+\Delta t)\, \psi(\boldsymbol{y},t+\Delta t) - 1\right|$
```

5. Results

5.1. Image Segmentation in 2D

Multi-Object Geodesic Active Contours (MOGAC) were applied to segmentation of 2D images into multiple compartments. The following segmentation example of a 512×512 “X” image was constructed to evaluate the performance of the algorithm.

In the first experiment, the image was segmented into 5 objects (i.e. compartments). Objects overlap in the initial segmentation for a total of $N = 8$ labeled object regions. The 5 objects were evolved with pressure and curvature forces [2]. Overlapping object regions were assigned a constant inward pressure force to cause their contraction. The final segmentation contains only the 5 objects with no gaps or overlaps (Fig. 2).

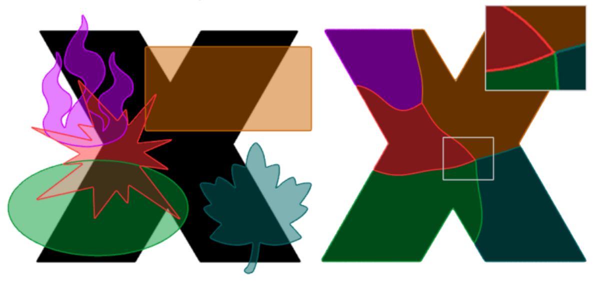


Figure 2: (a) Initial segmentation and (b) final segmentation.

To evaluate scalability of the algorithm, the problem size was increased by horizontally and vertically tiling the “X” image and initial segmentations. The algorithm was executed on a PC with dual Quad-core 2.54 GHz Intel Xeon processors and an NVIDIA Quadro 4000 graphics card. Fig. 3a shows the computation time per iteration averaged over the 2000 iterations required to segment each image. The original image took 4.9 sec. to segment on the GPU, and the computation time scaled almost linearly as a function of image size.

A second experiment was conducted to evaluate the algorithm’s performance as a function of the number of objects. Segmentation of the original image was initialized with between 1 and 16 randomly placed circles. Results are depicted in Fig. 3b. Computation time is almost constant as a function of the number of objects.

MOGAC was applied to cell tracking in a 1024×1024 microscopy image acquired from the Cell Centered Database (CCDB) [23]. The image was first automatically segmented with MIPAV [24] through gray-level morphology and thresholding techniques (Fig. 4a). The segmentation was then refined with MOGAC to better localize boundaries on the 138 detected objects (Fig. 4b). The segmentation was driven by pressure forces and external velocity field produced by GVF [25]. The GPU implementation of MOGAC ran for 250 iterations at a speed of 12 ms per iteration (3.03 sec. total).

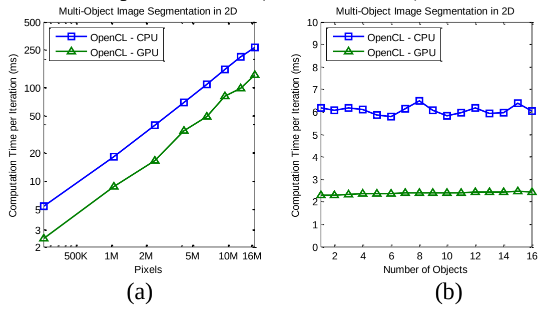


Figure 3: Segmentation of “X” image as a function of (a) image size and (b) number of initial objects.

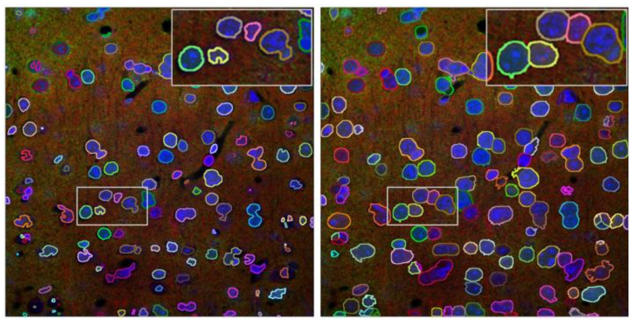


Figure 4: (a) Initial segmentation of microscopy image and (b) MOGAC segmentation refinement of 138 objects. Cells appear blue in this imaging modality.

5.2. Image Segmentation in 3D

The 3D version of MOGAC is almost identical to the 2D version, except that a 6-connected 3D neighborhood is used and finite difference calculations are evaluated on a $3 \times 3 \times 3$ stencil. As synthetic examples, MOGAC was applied to 3D segmentation of a metasphere and the Igea model at an image resolution of $256 \times 256 \times 256$.

In the first example, the segmentation was initialized with a torus and sphere which overlap to create 3 object regions. The segmentation process was again driven by pressure and curvature forces to produce a segmentation consisting of deformed versions of the torus and sphere without gaps or overlaps (Fig. 5). The segmentation required 700 iterations, and the GPU implementation required on average 196 ms per iteration (137 sec. total).

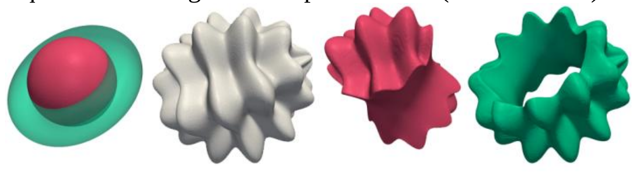
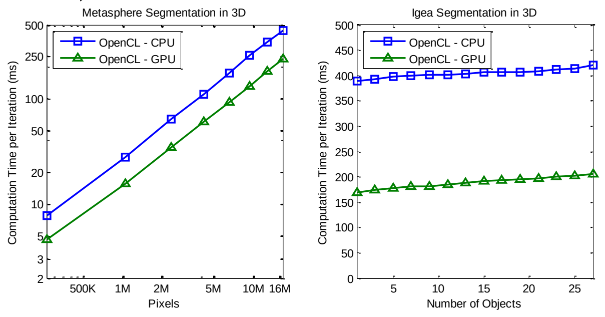


Figure 5: (a) Initial segmentation showing sphere and torus. (b) Target metasphere shape. Deformed (c) sphere and (d) torus.

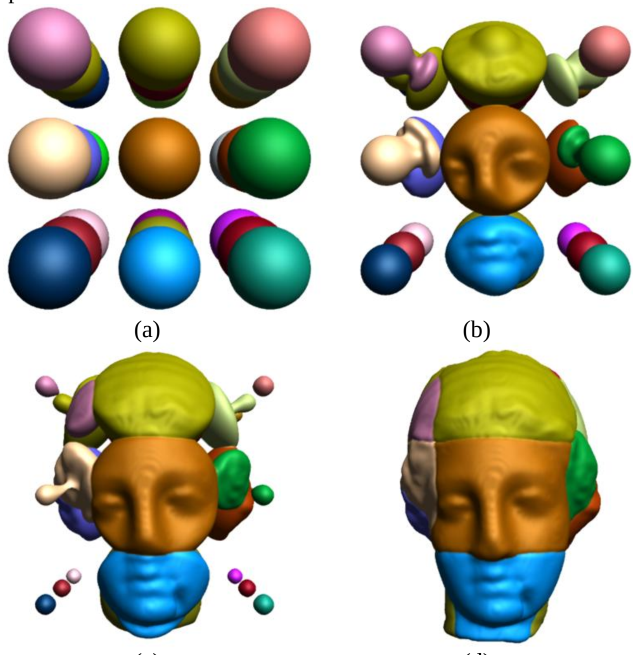
To evaluate the algorithm's scalability in 3D as a function of image size, the experiment was repeated on smaller images of size $64 \times 64 \times 64$ that were tiled horizontally and vertically. Results shown in Fig. 6a follow the same linear trends observed in the 2D case.

MOGAC was then applied to segmentation of the Igea model. The number of initial objects was varied between 1 and 27. Fig 6b depicts computation time as a function of the number of objects. Spheres placed outside the object contract to a point and disappear from the segmentation. The segmentation in Fig. 7 required 500 iterations, and the GPU implementation required 200 ms per iteration (100 sec. total).



(a) (b)

Figure 6: (a) Segmentation of Metasphere as a function of image size and (b) segmentation of the Igea model as a function of the number of objects. Computation time increases slightly in (b) because the total surface area increases with the number of spheres.

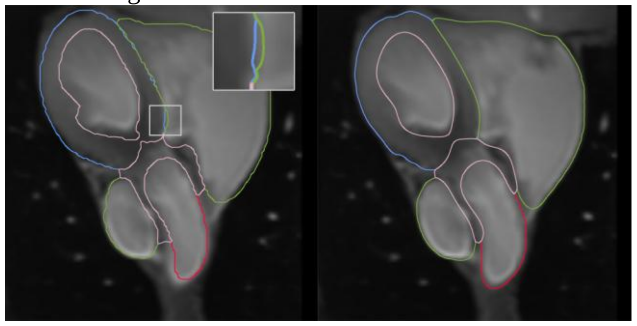


(a) (b) (c) (d)

Figure 7: Segmentation of Igea model from 27 objects showing (a) initial segmentation, (b) iteration 100, (c) iteration 200, and (d) iteration 500.

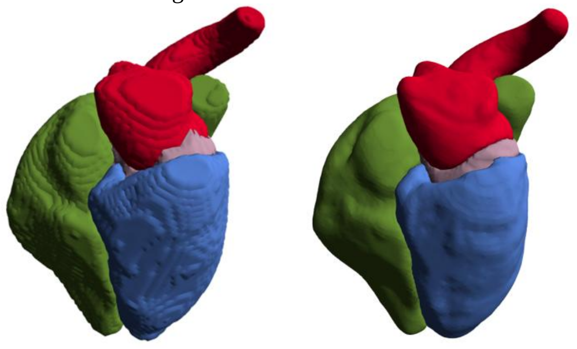
MOGAC was used to clean-up existing segmentations of MR images. In the first example, the algorithm was initialized with a previous segmentation of the epicardium in an $128 \times 256 \times 128$ MR image that contains small gaps and overlaps between structures (Fig. 8a). MOGAC was used to remove these gaps and overlaps to produce a sub-pixel segmentation that is a proper partition of the epicardium into 4 structures (Fig. 9b). The MOGAC clean-up required 50 iterations (2.5 sec. total).

In the second experiment, a whole brain was segmented into 10 structures with TOADS [26] on a $256 \times 256 \times 256$ MR image from the OASIS database [27]. The hard classification was smoothed with MOGAC for 10 iterations (2.3 sec. total) to produce the segmentation shown in Fig. 10b.



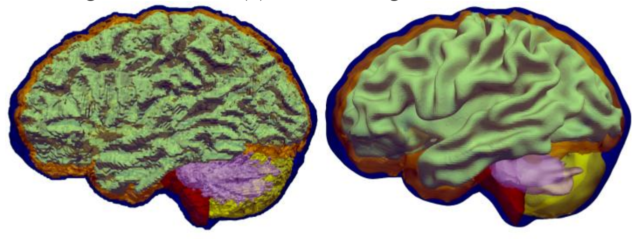
(a) (b)

Figure 8: Epicardium showing right ventricle and atrium (green), myocardium (blue), left ventricle (pink), and left atrium (red). (a) Initial heart segmentation and (b) MOGAC segmentation overlaid on MR Image.



(a) (b)

Figure 9: Epicardium showing right ventricle and atrium (green), myocardium (blue), left ventricle (pink), and left atrium (red). (a) Initial segmentation and (b) MOGAC segmentation.



(a) (b)

Figure 10: (a) Initial hard segmentation of 10 brain structures. (b) Segmentation after mean curvature smoothing with MOGAC.

6. Discussion

A Multi-object LSM has been presented that can segment any number of objects with the same small memory footprint. The algorithm was applied to image segmentation in 2D in 3D. The 2D version can run at 400 Hz for any number of objects at an image resolution of 512×512, making it applicable to real-time computer vision applications. The 3D version can segment images faster than the Multi-compartment LSM [9], which is reported to require 22.5 sec. per iteration to clean-up a segmentation of 10 brain structures at an image resolution of $256 \times 256 \times 205$. The same task was repeated on a $256 \times 256 \times 256$ MR image, for which MOGAC was approximately 100 times faster.

Table 1 summarizes the theoretical performance for each Multi-object LSM. MOGAC is theoretically faster and uses less memory than all existing methods. This is largely due to its use of the sparse-field instead of narrow-band method. Furthermore, implementing the algorithm in OpenCL provides additional speed-up by leveraging multiple cores available on the CPU or GPU.

Algorithm 1 spends a lot of computation time checking if $\exists n \text{ s.t. } \boldsymbol{x} \in \Gamma_n$. To avoid traversing the entire volume, a more work-efficient approach is to index Γ_n. A parallel algorithm for indexing Γ_n has already been described [16], and a variant of that algorithm is implemented in the open-source release of MOGAC.

TABLE I
ALGORITHM COMPLEXITY

Method	Time	Memory
N level set methods [4, 6, 8]	$O(NM^d \log M)$	$O(NM^d)$
Multi-phase [7]	$O(M^d \log M \log N)$	$O(M^d \log N)$
Multi-compartment [5, 9]	$O(M^d \log(M/N))$	$O(M^d)$
MOGAC	$O(M^d/P)$	$O(M^d)$
Work-efficient MOGAC[2]	$O(M^{d-1}/P)$	$O(M^d)$

Algorithm complexity for Multi-object LSMs based on M^d pixels, N objects, and P processing units in dimension $d = \{2,3\}$.

The algorithm's performance is tied to the sparse-field algorithm and its approximation to the signed distance field. It is known that this approximation has aliasing artifacts (see Fig. 7). To remove these artifacts, we recommend either smoothing the final iso-surface meshes in a post-processing step [28] or rendering objects with volumetric techniques that reduce the appearance of artifacts [29].

Segmentation problems can be phrased as either region growing, active contour segmentation, or a combination of both [30]. Active contour segmentation is analogous to simulation of physical objects that experience internal and external forces, whereas region growing is analogous to statistical classification of pixels that lie at the boundary of a region. Both interpretations have merit, but region growing is usually faster and requires less memory because it is only pixel accurate.

What we have presented is an efficient algorithm that makes multi-object level set segmentation competitive with region growing with respect to time and memory complexity. MOGAC is open-source and distributed as part of the Java Image Science Toolkit [19] to facilitate development of new segmentation algorithms that involve large numbers of objects.

[2] Although not described in this work, a variant of the work-efficient level set method [16] is implemented in the open-source release of MOGAC.

7. Acknowledgement

This work was funded by a graduate student fellowship from the Johns Hopkins Applied Physics Laboratory. We would like to thank Dr. Terry Peters at the Robarts Research Institute for providing Cardiac MRI data.

8. References

[1] J. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*: Cambridge Univ Pr, 1999.

[2] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International journal of computer vision,* vol. 22, pp. 61-79, 1997.

[3] D. Cremers, M. Rousson, and R. Deriche, "A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape," *International journal of computer vision,* vol. 72, pp. 195-215, 2007.

[4] C. Samson, L. Blanc-Féraud, G. Aubert, and J. Zerubia, "A level set model for image classification," *International journal of computer vision,* vol. 40, pp. 187-197, 2000.

[5] X. Fan, P. L. Bazin, and J. L. Prince, "A multi-compartment segmentation framework with homeomorphic level sets," presented at the Computer Vision and Pattern Recognition, IEEE Conf. on, 2008.

[6] N. Paragios and R. Deriche, "Coupled geodesic active regions for image segmentation: A level set approach," *Computer Vision—ECCV 2000,* pp. 224-240, 2000.

[7] L. A. Vese and T. F. 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.

[8] F. Losasso, T. Shinar, A. Selle, and R. Fedkiw, "Multiple interacting liquids," *ACM Transactions on Graphics (TOG),* vol. 25, pp. 812-819, 2006.

[9] X. Fan, P. L. Bazin, J. Bogovic, Y. Bai, and J. L. Prince, "A multiple geometric deformable model framework for homeomorphic 3D medical image segmentation," presented at the Computer Vision and Pattern Recognition Workshops (CVPRW '08), 2008.

[10] R. Adams and L. Bischof, "Seeded region growing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 16, pp. 641-647, 1994.

[11] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 22, pp. 888-905, 2000.

[12] K. Li, E. D. Miller, M. Chen, T. Kanade, L. E. Weiss, and P. G. Campbell, "Cell population tracking and lineage construction with spatiotemporal context," *Medical image analysis,* vol. 12, pp. 546-566, 2008.

[13] J. Lie, M. Lysaker, and X. C. Tai, "A binary level set model and some applications to Mumford-Shah image segmentation," *Image Processing, IEEE Transactions on,* vol. 15, pp. 1171-1181, 2006.

[14] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences,* vol. 93, p. 1591, 1996.

[15] R. Whitaker, "A level-set approach to 3D reconstruction from range data," *International journal of computer vision,* vol. 29, p. 231, 1998.

[16] M. Roberts, J. Packer, M. C. Sousa, and J. R. Mitchell, "A work-efficient GPU algorithm for level set segmentation," in *HPG '10* 2010, pp. 123-132.

[17] G. Tornai and G. Cserey, "2D and 3D level-set algorithms on GPU," 2010, pp. 1-5.

[18] H. Mostofi, J. Schnabel, and V. Grau, "Fast Level Set Segmentation of Biomedical Images using Graphics Processing Units," Oxford University2009.

[19] B.C. Lucas, J.A. Bogovic, A. Carass, P.L. Bazin, J.L. Prince, D. Pham, and B. A. Landman, "The Java Image Science Toolkit (JIST) for Rapid Prototyping and Publishing of Neuroimaging Software," *Neuroinformatics*, 2010.

[20] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM Siggraph Computer Graphics,* vol. 21, pp. 163-169, 1987.

[21] R. Courant, K. Friedrichs, and H. Lewy, "On the partial difference equations of mathematical physics," *IBM Journal of Research and Development,* vol. 11, pp. 215-234, 1967.

[22] M. Harris, G. E. Blelloch, B. M. Maggs, N. K. Govindaraju, B. Lloyd, W. Wang, M. Lin, D. Manocha, P. K. Smolarkiewicz, and L. G. Margolin, "Optimizing parallel reduction in CUDA," *Proc. of ACM SIGMOD, 21,* vol. 13, pp. 104-110.

[23] M. E. Martone, A. Gupta, M. Wong, X. Qian, G. Sosinsky, B. Ludäscher, and M. H. Ellisman, "A cell-centered database for electron tomographic data," *Journal of Structural Biology,* vol. 138, pp. 145-155, 2002.

[24] M. J. McAuliffe, F. M. Lalonde, D. McGarry, W. Gandler, K. Csaky, and B. L. Trus, "Medical image processing, analysis & visualization in clinical research," *cbms,* p. 0381, 2001.

[25] X. Chenyang and J. L. Prince, "Snakes, shapes, and gradient vector flow," *Image Processing, IEEE Transactions on,* vol. 7, pp. 359-369, 1998.

[26] P. L. Bazin and D. L. Pham, "Topology-preserving tissue classification of magnetic resonance brain images," *Medical Imaging, IEEE Transactions on,* vol. 26, pp. 487-496, 2007.

[27] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open Access Series of Imaging Studies (OASIS): cross-sectional MRI data in young, middle aged, nondemented, and demented older adults," *Journal of Cognitive Neuroscience,* vol. 19, pp. 1498-1507, 2007.

[28] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *ACM Transactions on Graphics (TOG),* vol. 22, pp. 950-953, 2003.

[29] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross, "Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces," 2005, pp. 303-312.

[30] S. C. Zhu and A. Yuille, "Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 18, pp. 884-900, 1996.