Unifying Deformable Model Representations through New Geometric Data Structures

Blake C. Lucas

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland
July, 2012

© Blake C. Lucas 2012
All rights reserved
Abstract

One objective of image analysis is to extract geometric information from images. A modern approach is to start with an initial guess for an object's geometry and deform it to match objects observed in images. Deformable models are commonly used to represent geometry because they change shape due to forces applied at their boundary. In image analysis, deformable models imitate all types of materials: rigid, elastic, plastic, and fluid materials. Meshes and level sets are the two primary deformable model representations. Methods usually favor a particular representation depending on the type of material the model is intended to imitate. For instance, we will describe a new algorithm for 3D reconstruction (SxMAC) with a strong preference for level sets. However, image analysis pipelines that use a mixture of methods are forced to transform one representation into another in order to use the preferred representation for each method. This strategy leads to loss of information and less flexibility in design of the system.

Spring Level Sets (SpringLS) merge meshes and level sets into a single representation to provide interoperability between methods designed for either. The key idea is to use a constellation of disconnected triangular surface elements (springls) to define a level set. SpringLS is extended to the multi-object case by coupling it with Multi-Object Geodesic Active Contours (MOGAC). The MOGAC data structure can represent any number of level sets with a label mask and distance field. The combination of SpringLS and MOGAC creates a powerful new type of deformable model (MUSCLE) for representing multiple objects that is parametric, has sub-voxel precision, re-meshes, tracks point correspondences, and guarantees no self-intersections, air-gaps, or overlaps between adjacent structures. Applications to full brain parcellation, whole body segmentation, atlasng, and multi-organ tracking are presented.

Finally, a new method is described for warping and combining images in ways that preserve edge strength. It does so by decoupling shape and intensity information, manipulating them independently, and then synthesizing images. The method relies on MUSCLE to represent geometry in an image and enforce geometric constraints (e.g. no air-gaps or overlaps). Applications to registration and atlas construction are presented.
Thesis Committee

**Primary Advisor:** Dr. Russell H. Taylor, John C. Malone Professor

Department of Computer Science

The Johns Hopkins University

**Secondary Advisor:** Dr. Michael Kazhdan, Associate Professor

Department of Computer Science

The Johns Hopkins University

**Reader:** Dr. Jerry L. Prince, William B. Kouwenhoven Professor

Department of Electrical and Computer Engineering

The Johns Hopkins University
Acknowledgments

I thank Prof. Russ Taylor, Prof. Michael Kazhdan, and Prof. Jerry Prince, all of whom have served as my graduates advisors over the past several years and have been instrumental in guiding this work and my professional development. I thank Dr. Taylor for providing a context and direction for this work along with his insightful criticisms and contributions that enhanced the clarity and impact of this work. I thank Dr. Kazhdan for his thorough critiquing of the principles and mathematics behind this work as well as his assistance in differentiate this work from prior art in the broad realm of computer graphics and geometric modeling. I thank Dr. Prince for providing me with extensive training in medical analysis, especially deformable models, which was foundational to this line of research and motivated many of the contributions described in this work.

I thank Dr. Mehran Armand at the JHU Applied Physics Laboratory for his financial support and technical supervision of this research. I also thank him for permitting me the freedom to explore a broad set of topics, without this work would not be possible.

I thank my fellow lab members in the Laboratory for Computational Sensing and Robotics (LCSR) and Biomedical and Image-Guided Surgical Systems (BIGSS) whom I have collaborated with and learned from over the last several years.

I thank Dr. Todd McNutt, Dr. Ted Deweese, Dr. Lee Myers, Dr. Ben Tsui, Dr. George Fung, Dr. Terry Peters, and Dr. John Carrino for assistance in obtaining imaging data.

I thank Dr. Hamid Krim at N.C. State University for giving me an opportunity as an undergraduate to gain experience in image analysis which lead to my pursuit of a Ph.D.

I thank my parents, Michael and Kimberly Lucas, as well as my sister Claire for their relentless support of my educational pursuits.
# Contents

1 Introduction .................................................................................................................. 1  
  1.1 Motivation ................................................................................................................ 1  
  1.2 Contributions ........................................................................................................... 4  
  1.3 Outline ...................................................................................................................... 6  
2 Background .................................................................................................................... 7  
  2.1 Image Analysis .......................................................................................................... 7  
    2.1.1 Image Pipelines .................................................................................................. 7  
    2.1.2 Global Registration ......................................................................................... 7  
    2.1.3 Deformable Registration ................................................................................ 8  
    2.1.4 Image Processing ............................................................................................... 9  
    2.1.5 Classification .................................................................................................... 10  
    2.1.6 Localization ...................................................................................................... 10  
    2.1.7 Segmentation .................................................................................................... 10  
    2.1.8 Mapping and Tracking ..................................................................................... 11  
    2.1.9 Atlas Construction ......................................................................................... 11  
    2.1.10 Comprehensive Imaging Pipeline (CIP) ....................................................... 12  
  2.2 Geometric Models .................................................................................................... 13  
    2.2.1 Triangle Meshes ............................................................................................... 13  
    2.2.2 Tetrahedral Meshes ......................................................................................... 14  
    2.2.3 Label Masks ..................................................................................................... 16  
    2.2.4 Level Sets ........................................................................................................ 16  
    2.2.5 Particle Systems .............................................................................................. 17  
    2.2.6 Currents ........................................................................................................... 19  
    2.2.7 Complete Distance Field Representation (CDFR) ..................................... 19  
    2.2.8 Other Representations .................................................................................... 20  
  2.3 Appearance Models .................................................................................................. 23  
  2.4 Summary .................................................................................................................. 24  
3 Reconstruction from C-arm X-ray Images .................................................................... 25  
  3.1 Motivation ................................................................................................................ 25  
  3.2 Background .............................................................................................................. 26  
  3.3 Method .................................................................................................................... 31  
  3.4 Results .................................................................................................................... 35  
    3.4.1 Phantom Experiments ...................................................................................... 35  
    3.4.2 SxMAC Reconstruction with Synthesized Images ....................................... 38  
    3.4.3 SxMAC Reconstruction with Real Images ................................................... 44  
  3.5 Discussion ................................................................................................................ 47  
  3.6 Conclusion ............................................................................................................... 49  
4 Multi-Object Geodesic Active Contours .................................................................... 50  
  4.1 Motivation .............................................................................................................. 50  
  4.2 Background ............................................................................................................ 50  
  4.3 Overview ............................................................................................................... 51
Chapter 1

1 Introduction

1.1 Motivation

One objective of image analysis is to extract geometric information from images. Examples include construction of statistical atlases, analysis of cross-sectional differences, analysis of longitudinal changes, and computer-aided diagnosis. A modern approach is to start with an initial guess for an object's geometry and deform it to match objects observed in images. Geometric information can be extracted more accurately if the initial guess is close to the actual object's geometry and pose. There are a variety of image analysis methods for deforming geometric models to match images and for guessing an object's geometry based on prior knowledge. Image analysis methods are sub-divided into several core tasks: localization, global registration, deformable registration, segmentation, tracking, and atlassing. This work will focus on global registration, deformable registration, and segmentation. Conceptually, the distinction between these three tasks is the type of material the shape model is intended to imitate. Figure 1 depicts the spectrum of materials that are imitated in imaging applications. For example, global and deformable registration model elastic materials while active contour segmentation models viscoelastic and fluid materials.

![Materials Continuum](image)

*Figure 1: The materials continuum [1]. In imaging applications, deformable models imitate the behavior of materials between elastic and plastic (shown in green). Also shown is the threshold between elastic and plastic materials where re-meshing becomes a requirement for mesh-based representations.*
We use the word "imitate" because the behavior of deformable models does not have to be physically accurate. In fact, physical accuracy is almost impossible because material properties and boundary conditions are rarely known beforehand. Moreover, relaxing the constraint that the model be physically accurate allows us to write faster algorithms that break the laws of physics (e.g. mass conservation). In this non-physical context, the materials continuum is understood in terms of Degrees of Freedom (DOF). Elastic deformations are smooth, low DOF (tens to thousands) transformations while fluid-like deformations are less smooth, high DOF (thousands to millions) transformations.

The extent to which images can be analyzed is controlled by two main factors: image contrast and shape variability. Contrast is classically defined as either the difference or ratio of image intensities that makes boundaries between objects identifiable [2], but that definition can be extended to any measurable differences (e.g. statistical distributions, texture features, colors, etc.). Objects naturally have variability in shape. A primary objective of image analysis is to characterize this variability. Some amount of shape variability is usually known a priori through manual studies and/or high-fidelity, high-contrast imaging techniques. This prior knowledge can be used to build statistical shape models (atlases) that augment the software's ability to find object boundaries in regions with poor image contrast. Table 1 describes examples of applications and the relative shape variability and image contrast associated with each.

### Table 1: Image analysis applications and their relative shape variability and image contrast.

<table>
<thead>
<tr>
<th></th>
<th>Poor Contrast</th>
<th>Good Contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Little Shape Variability</strong></td>
<td>Pre-operative CT to ultrasound registration</td>
<td>Lung(^1) tracking in 4D CT</td>
</tr>
<tr>
<td><strong>Some Shape Variability</strong></td>
<td>Liver segmentation in CT and sub-cortical segmentation in MRI</td>
<td>Lung(^1) segmentation in CT and brain ventricle segmentation in MRI</td>
</tr>
<tr>
<td><strong>High Shape Variability</strong></td>
<td>Tumor segmentation in mammograms &amp; bone cement reconstruction from C-Arm X-Ray images.</td>
<td>Cortical surface reconstruction in MRI &amp; adipose (fat) segmentation in CT</td>
</tr>
</tbody>
</table>

\(^1\) In the context of this work, lung refers to the exterior contour of the left and right lung, which undergoes elastic (low DOF) deformations during respiration. Unlike the exterior structure, the lung's interior structure is highly variable between subjects and will not be discussed in this work.

To solve such image analysis problems, the common approach is to start with low DOF methods, such as rigid registration, and then gradually increase the DOFs to use non-rigid deformable
registration and eventually fluid-like active contour segmentation. This pipelined approach produces more accurate results and allows the system to be iteratively refined and tuned to address challenges specific to the imaging scenario and analysis objectives. Because high DOF methods are more sensitive to image contrast, they should not be used in all imaging applications. Table 2 outlines the types of methods that should be used depending on shape variability and image contrast.

Table 2: Outline of image methods that should be used depending on image contrast and shape variability for a particular application. For definitions of methods, see section 2.1.

<table>
<thead>
<tr>
<th>Shape Variability</th>
<th>Poor Contrast</th>
<th>Good Contrast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>Global Registration + Deformable Registration</td>
<td>Global Registration + Deformable Registration</td>
</tr>
<tr>
<td>Variability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Some</td>
<td>Global Registration + Atlas-based Registration</td>
<td>Global Registration + Deformable Registration or Atlas-based Segmentation</td>
</tr>
<tr>
<td>Variability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Localization with Machine Learning + Segmentation</td>
<td>Global Registration + Deformable Registration + Segmentation</td>
</tr>
<tr>
<td>Variability</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There are a variety of geometric data structures used to model different types of materials. Triangle meshes, level sets, and label masks are the three most popular in image analysis. However, each one has different strengths and weakness, and there is usually a preferred representation for each imaging task.

We will present one application for which level sets have significant advantages over meshes. The Sparse X-Ray Multi-view Active Contour (SxMAC) algorithm will motivate the need to have a level set representation in image analysis. One reason is that level sets make it easy to simulate the forward imaging process because they provide a fast test to determine if a voxel is inside or outside an object. After introducing new data structures for combining meshes and level sets, we will present a Parametric Image (p-image) representation that uses the fast inside / outside test provided by level sets to synthesize MR and CT images. The ability to quickly synthesize images will lead to new atlasing methods that model variation in geometry and image intensity information simultaneously.
1.2 Contributions

The work described in this thesis merges triangle meshes, level sets, and label masks into a single geometric data structure that preserves the strengths of each. The new data structure provides interoperability between imaging tasks that previously were difficult to combine (e.g. intensity-based image registration and active contour segmentation). A number of examples will be presented that demonstrate geometric properties and versatility of the data structure in the context of image analysis. Before describing the final data structure, two new data structures will be introduced that are used to create the third (Figure 2). Source code is available at http://code.google.com/p/imagesci/. The following is a list of major contributions described in this work:

1) Sparse X-ray Multi-view Active Contours (SxMAC) motivate the need for a level set representation in image analysis. The algorithm requires a geometric data structure that can easily change topology, be used to simulate the forward imaging process, and mask image volumes. The level set representation is well suited for these tasks.

2) Multi-Object Geodesic Active Contours (MOGAC) compress multiple level sets into a distance field and label mask. We will describe how to represent multiple objects with the "label mask + distance field" data structure and evolve them simultaneously with active contour methods. The new algorithm has lower computational complexity and uses less memory than all existing multi-object level set methods.

3) Spring Level Sets (SpringLS) merge level sets and triangles meshes into one data structure. The key idea is to remove connectivity from triangle meshes and replace it with attraction forces that provide soft connectivity between triangles. Because the mesh is disconnected, it can be re-sampled / re-meshed quickly and simultaneously with an evolving level set to imitate fluid-like behavior. SpringLS can be used to model elastic and fluid-like materials with the same representation, which we demonstrate is useful for atlas based segmentation.

4) Multi-object Spring Level Sets (MUSCLE) is a combination of MOGAC and SpringLS. The "springls + label mask + distance field" data structure has sub-voxel precision, is parametric, re-
meshes, tracks point correspondences, and guarantees no self-intersections, air-gaps, or overlaps between adjacent structures. Numerous examples will be presented that use MUSCLE to register, segment, track, and atlas multiple objects in CT and MR images.

5) **Parametric Images (p-images)** attach an image intensity model to the MUSCLE representation to describe both shape and appearance information in an image. Since shape and appearance information are decoupled, geometric information can be warped and combined independently of intensity information. The main advantage of this representation is that it preserves edge strength when images are warped and combined during atlas construction. Applications to atlas construction with MR and CT images will be presented.

![Deformable Models Diagram](image)

**Figure 2:** In this work, deformable models used for image analysis (triangle meshes, level sets, and label masks) will be combined to create new types of deformable models (SpringLS, MOGAC, and MUSCLE).
1.3 Outline

- **Chapter 2** reviews the methods and geometric data structures used in image analysis. Geometric properties, strengths, and weakness of different representations will be discussed. These discussions will motivate subsequent chapters on new representations.

- **Chapter 3** motivates the continued need for level sets in image analysis. A novel algorithm is presented for reconstructing objects from several X-Ray images that highlights capabilities of level sets including: topology changes, imaging simulation, and image masking.

- **Chapter 4** introduces Multi-Object Geodesic Active Contours (MOGAC) and the "label mask + distance field" data structure. A parallel algorithm is described for segmenting images with MOGAC. Results are presented that demonstrate the scalability and computational performance of the algorithm on the CPU and GPU.

- **Chapter 5** presents the Spring Level Set (SpringLS) representation. Mathematical foundations and deformation algorithms are described along with details about its implementation. Examples of elastic and fluid-like deformations are presented with applications to image segmentation and registration.

- **Chapter 6** integrates SpringLS and MOGAC to create a new data structure (MUSCLE) for representing multiple objects that has numerous important geometric properties. Methods and examples are presented for using the data structure to perform tracking, segmentation, and atlasing tasks.

- **Chapter 7** attaches image intensity information to the MUSCLE representation to characterize both shape and appearance information in an image. Using the parametric image representation, a new method is presented for warping and combining images in a way that preserves edge strength. Applications to atlas construction with MR and CT images are presented.

- **Chapter 8** concludes our discussion and summarizes the major contributions of this work and directions for future research.
Chapter 2

2 Background

2.1 Image Analysis

2.1.1 Image Pipelines

A pipeline is a feed-forward system of tasks in which the output from one task is the input to one or more other tasks. Tasks are conceptually autonomous, allowing the underlying algorithms to be encapsulated and interfaced with a small number of input and output data structures. The input and output parameters usually include images and / or data structures representing objects in those images. Tasks are cascaded to create long chains of processing steps. Because tasks are autonomous and loosely coupled, pipelines can be reconfigured to achieve different objectives. The extent to which pipelines can be reconfigured is limited by the compatibility of output parameters to input parameters (i.e. image outputs can connect to image inputs, but triangle mesh outputs cannot connect to level set inputs).

If output and input data structures are incompatible but represent similar information, then tasks are added to transform one data structure into another. For instance, a triangle mesh can be rasterized to a level set, and an iso-surface can be extracted from a level set to create a triangle mesh. These data structure transformation tasks lose information, which reduces accuracy and flexibility in design of the system. We now review the components of image pipelines and the data structures used by each.

2.1.2 Global Registration

Global registration refers to a low DOF transform of 3D image space that puts one image into alignment with another. Examples include rigid (6 DOF), rigid + scale (7 DOF), rigid + non-uniform scale (9 DOF), and affine (12 DOF) transformations [3, 4]. A popular method for globally registering two images is FLIRT [5]. The inputs to FLIRT are two images (source and target), and
the outputs are the transformed source image and 4x4 transformation matrix \( A \). Another approach to global registration is Iterative Closest Point (ICP) [3]. The inputs to ICP are two triangle meshes or a triangle mesh and a point cloud representing two objects found in the source and target images. The outputs are a transformed triangle mesh and 4x4 transformation matrix \( A \). ICP requires a prior 3D surface model of an object and a method for locating closest points on a corresponding surface in the image. Both approaches, FLIRT and ICP, have important applications in imaging and either can be used in certain imaging scenarios. However, FLIRT and ICP perform very differently, and one is not an equal substitute for the other. Note that in order to replace FLIRT with ICP in a pipeline (or vice versa), data structure transformations must be introduced on the input and output of the registration algorithm, which adds overhead to the pipeline.

2.1.3 Deformable Registration

Deformable image registration [6-9] has become a popular technique for reliably and automatically segmenting multiple objects with little prior knowledge of anatomy or imaging technology. The idea is to define a sparse set (hundreds to thousands) of control points and specify where they map to in a target image. The ambient space is then warped to put the control points and underlying images into alignment. There are non-rigid extensions to ICP that perform a similar non-linear warping based on mesh representations of objects [10]. Like FLIRT and ICP, there are situations where non-rigid ICP can replace deformable image registration in an imaging pipeline. However, we will limit our discussion to deformable image registration in this work.

To overcome a registration algorithm's lack of prior knowledge, there is a strong assumption that the target's anatomy is a smooth elastic deformation of the source's anatomy. This is rarely the case if the source and target come from different subjects, leading to what is known as "atlas bias" where the warped source image still resembles the source more than the target. However, deformable registration does provide a good initialization for subsequent segmentation algorithms in the image analysis pipeline.
The output of the registration algorithm is a displacement field describing where voxel locations in the target image map to in the source image. This 3D-to-3D mapping allows any geometric structures identified in the source image to be warped into the target image. Note that displacement fields produced by popular registration algorithms (e.g. [6, 7, 11-13]) are not invertible or inverses of each other, which is to say composing displacement fields obtained from source-to-target and target-to-source registration does not produce an identity mapping. This will become a concern later when trying to deform meshes with source-to-target displacement fields and images with target-to-source displacement fields.

2.1.4 Image Processing

Image processing is a type of digital signal processing that is specific to images. For a survey of methods, see Woods and Gonzalez [14]. Image processing is a pre-requisite for segmentation because segmentation algorithms are sensitive to initialization and image contrast. One objective of image processing is to remove noise. Many imaging devices have very little signal noise. In medical imaging, structured noise is more common and also more difficult to remove. Structured noise is due to the image reconstruction algorithm, acquisition technology, or physics of the imaging modality. Examples include shading artifacts in MR images due to inhomogeneities in the main magnetic field, metal streak artifacts that occur in CT images due to strong X-ray attenuation, and speckle noise in ultrasound images due to interference patterns produced by reflected sound waves. Structured noise is more difficult to model statistically, which has lead to development of non-linear morphological filters that are specific to imaging [15-17] among other methods [18, 19].

Once major sources of noise have been removed, remaining uncertainty in the shape and location of objects is due to contrast, shape variability, occlusion, and the confusion between objects of interest and objects not of interest. Instead of grouping objects not of interest into a single background region; in this work, we will try to find all objects. For certain images, this could increase the problem size to hundreds if not thousands of objects. To address this challenge, we will present new data structures that mitigate the computation and memory overhead associated with segmenting many objects.
2.1.5 Classification

As previously mentioned, the definition of image contrast is extendable to any measurable differences that can be derived from image intensities. These measurable differences are used to classify regions of an image as part of a particular object [20-23]. During classification, the probability or likelihood that a voxel is inside an object is estimated at each voxel to produce a gray-level image (likelihood image) whose contrast is consistent with the classic definition (i.e. the difference or ratio of image intensities that makes boundaries between objects identifiable). Instead of a hard assignment of each voxel to a particular object, many algorithms produce a soft or fuzzy classification that indicates the likelihood that a voxel belongs to each object. The classification result is then fed into a segmentation system that extracts geometry from the likelihood image. If the statistical model is easy to compute, fuzzy memberships can be iteratively re-estimated during image processing, registration, or segmentation process.

2.1.6 Localization

Localization refers to detecting whether an object is present in an image and estimating its approximate location as either a single point, bounding box, or pose matrix. The problem is formulated as a classification or matching problem and solved with machine learning methods [24]. To do so, salient features are extracted from images to construct descriptors [25] which are matched against descriptors of known geometries. Once matched, the location and pose of the object are estimated. Localization usually requires manually segmented or labeled image data to train and tune the classifier. Localization is important because many segmentation methods are sensitive to the initial estimate for an object’s shape and/or pose.

2.1.7 Segmentation

Image segmentation is the process of extracting geometry from structures observed in images. Classically, the problem was formulated as the process of labeling image pixels by threshold techniques or region-based split and merge techniques [26]. The output of these systems is referred to as a label mask, which identifies each pixel as either a member of an object or part of the background. The definition of segmentation has been extended to seeded region growing [27], graph cuts [28], parametric methods [29], and level set methods [30]. Parametric methods
and level set methods both have sub-pixel precision. These high-precision methods start with some initial geometric model and then deform the model to align it with boundaries visible in an image. The output of these systems is either a triangle mesh or signed distance field. Segmentation systems are designed for specific imaging applications and are often a combination of several segmentation methods. However, methods designed for one geometric representation are not necessarily transferable to a different geometric representation. This lack of interoperability between methods for different geometric representations will be addressed in this work.

2.1.8 Mapping and Tracking

One way to map or track objects is to identify where locations in one image map to in another image. The term "tracking" is used if the images are of the same object at different time points after undergoing some type of motion [31], and "mapping" refers to images of different objects [32, 33] or the same object that has been physically altered [34] or imaged with a different technology [35]. The same techniques can be used for either, but tracking is usually the easier of the two problems. Mapping or tracking information is represented either as a corresponding set of points (e.g. vertices on a triangle mesh) in source and target images, or a displacement field which describes where every location in the target image maps to in the source image. These representations are not equivalent because meshes provide a boundary (surface) mapping whereas displacement fields provide a volumetric mapping for all points in image space. This work will address how to combine triangle meshes and level sets to provide both volumetric and surface mappings.

2.1.9 Atlas Construction

An atlas is a statistical representation of a geometric structure based on several shape instances reconstructed from images. Atlas construction requires both segmentation and mapping. For a survey of methods, see Heimann and Meinzer [36]. Point Distribution Models (PDMs) are very popular in medical imaging [37, 38]. PDMs take as input a collection of several shape instances represented by triangle meshes that have matching vertex correspondences. Principal Component Analysis (PCA) is performed on all the vertices to represent the structure's variation
with a high dimensional Gaussian distribution (i.e. a mean shape paired with mode weights for each vertex). The concept of PCA on shapes has been extended to level sets [39], and deformation fields [12]. Although these methods are all designed to represent shape variation, they are not equivalent because PCA is used in different shape spaces. One known problem with PDMs is that shape instances generated by the atlas may have invalid geometry. For instance, triangle surfaces may self-intersect. A related problem occurs if level sets are used. Level sets cannot self-intersect, but they can change topology. It is possible that shape instances generated from a level set atlas can have a different topology than the true structure or any shape instance used to build the atlas. Examples will be presented in Chapter 6 that demonstrate how to enforce geometric constraints when generating shape instances from Point Distribution Models.

### 2.1.10 Comprehensive Imaging Pipeline (CIP)

One incarnation of a fully automated imaging pipeline for localization, registration, segmentation, tracking, and atlas construction is depicted in Figure 3.

![Diagram of Comprehensive Imaging Pipeline (CIP)](image)

**Figure 3:** Concept of a Comprehensive Imaging Pipeline (CIP) that localizes, registers, segments, and tracks objects. It then uses the segmented objects to iteratively refine the atlas and analyze intra / inter subject variability of geometric structures. PDM refers to Point Distribution Model [37], although any parametric atlas representation could be used in its place. Shape information is passed between modules in the form of a descriptor, geometry, or statistical atlas.

Systems that implement large sections of a Comprehensive Imaging Pipeline (CIP) are emerging in literature, along with software infrastructures for building them [40-43]. In the work by Kohlberger et al. [44], they begin with machine learning followed by mesh-based atlas registration and conclude with level set active contour segmentation. In the transition from mesh to level set representation, they lose point correspondences between the deformable
model and atlas. This impedes their ability to progress further through the CIP and refine their atlas. In comparison, Tosun et al. [45] implement the second half of the CIP. They begin with an active contour segmentation of the human brain cortex and then parametrically map the isosurface to an atlas. The drawback of their system is that the parametric atlas they construct cannot be fed back into their active contour segmentation, which uses an implicit level set representation. This work presents a new type of deformable model that can be used to implement all stages of the CIP. The following section reviews existing deformable model representations and drawbacks.

2.2 Geometric Models

2.2.1 Triangle Meshes

Meshes (Figure 4) were the earliest representation for deformable models [29]. In this framework, the model is deformed by perturbing mesh vertices. The model's boundary is explicitly tracked by remembering the trajectory of each vertex. As a section of the mesh expands or contracts, sharp creases, edges, self-intersections, or triangle flips can develop. Sharp edges and other mesh artifacts violate a common material property that objects represented by deformable models are between elastic and plastic on the materials continuum (Figure 1). To reduce artifacts, the mesh must be regularized and re-sampled (re-meshed) periodically. Because mesh triangles must be connected to form a watertight model, re-meshing is challenging [46-52] and interferes with vertex tracking in non-intuitive ways. Re-meshing becomes more challenging if the mesh is not allowed to self-intersect, is allowed to change topology, or shares a boundary with another object. For these reasons, triangle meshes have become unpopular for applications where the model 1) undergoes large non-rigid deformations that require re-meshing; 2) the model is expected to change topology; 3) the model is likely to self-intersect.
Figure 4: Igea model represented as a triangle mesh. Inset shows triangles. Data courtesy of Aim@Shape (http://shapes.aimatshape.net/).

2.2.2 Tetrahedral Meshes

Tetrahedral or "tet" meshes can be used to represent boundary and volumetric information for structures (see Figure 5). Finite Element Methods (FEM) can be used to simulate physical behavior of structures represented by tetrahedral meshes [53, 54]. Methods exist for automatically creating tetrahedral meshes from triangle meshes via constrained Delaunay triangulation [55-57]. After which, tetrahedra can be tracked during deformation to provide an invertible surface and volumetric mapping between source and target geometry. Tetrahedral methods have been used for simultaneous segmentation and registration of the Pelvis [58] and Brain [59, 60]. The difficulty with tetrahedral representations is that in order to preserve an invertible mapping, tetrahedra must not be allowed to invert during deformation. Tetrahedra can easily invert if the model is significantly compressed or warped [61]. One approach to avoiding this scenario is to use large tetrahedrons. Large tets reduce the precision and accuracy of the model, so other strategies are needed to prevent tet inversion. A second approach is to detect when a tet inversion is about to occur and prevent the tet from deforming. This strategy adds drag to the deformation and can create tets with poor aspect ratios, which prevents
convergence of simulations that use Finite Elements. It is typical to add a smoothing operation to maintain tet quality and hopefully remove tetrahedron inversions. However, smoothing tet meshes does not guarantee the number of inversions decreases and can even create new inversions, so neither tetrahedra quality nor topology are enforceable with smoothing.

To model invertible deformations in Finite Element simulations, it is common to manually or semi-automatically edit tetrahedral meshes until one is found that produces physically accurate simulation results. This approach is not an option in imaging applications where it is expected that methods are completely automated and robust to a range of deformations and initial conditions. If an invertible mapping is not needed, then there are online strategies for re-meshing that improve tetrahedron quality [49, 62]. Another option is to embed a surface mesh in a coarse volumetric mesh [1, 63, 64] and compute physical simulations on the coarse volumetric mesh. The deformation of volumetric elements is used to drive the deformation of embedded surface elements. If the deformation is fluid-like, then the entire volume is re-meshed without changing the embedded surface mesh, which preserves surface mapping but not volumetric mapping. These methods are robust to large deformations and guarantee tetrahedron quality, but do not maintain an invertible volumetric mapping. However in imaging, the invertibility requirement must be relaxed to build general purpose tools because there are many important imaging scenarios that are not invertible (i.e. articulated joints or any scenario with sliding surfaces, truncated CT scans, pre to post-operative image registration).

![Tetrahedral mesh for a manifold object rendered with NetGen](http://www.hpfem.jku.at/netgen/)
2.2.3 Label Masks

It is common in image analysis for objects to be first manually segmented with a painting tool to produce a binary mask for each object [65]. These masks are then merged together to create a label mask image (see Figure 6). Displacement fields can be applied to labels by warping and re-sampling the label mask image. Since label image values are region indicators, only nearest-neighbor interpolation can be used. Label masks have only voxel precision and have no ability to maintain point correspondences or surface labels; however, they are very popular because they can be stored and manipulated with a smaller memory footprint than maintaining independent level sets (see section 2.2.4).

Since most ground-truth segmentations are represented as label masks, new image analysis methods must have a way of converting their geometric representation to a label mask for evaluation. There has been less emphasis on sub-voxel precision geometric representations because local sub-voxel differences are undetectable with label masks. We believe sub-voxel precision is very important because a lot of geometric information is embedded in the normals and curvature of a shape, which cannot be accurately represented with a label mask. In Chapter 4, we will present examples of cortical surface segmentation where the difference between voxel and sub-voxel accuracy manifests as dramatic differences in shape.

![Figure 6: Label mask for human brain cortex. Colors identify gyri.](image)

2.2.4 Level Sets

The level set method [66, 67] represents a deformable model as a 3D function of space represented by a image. The signed distance field is common choice for the 3D function; in
which, the image intensity at each voxel is a distance measurement to the surface of the object (see Figure 7). Distance measurements are signed: negative values are inside the object and positive values are outside. A triangle mesh can be extracted by computing the iso-surface corresponding to the zero level set of the image. The level set representation has several advantages over deformable meshes: 1) There is no need for self-intersection removal; 2) topology change is easy; 3) There is no need to re-mesh. These properties have made level sets the popular choice for image segmentation and fluid-like non-rigid deformation. They are also excellent for modeling multiple objects that share boundaries. For example, several deformation schemes enforce that adjacent objects represented by level sets cannot overlap or have air gaps [44, 68, 69].

![Figure 7: Level set function $\varphi: \mathbb{R}^{3} \rightarrow \mathbb{R}$. (a) cow iso-surface. (b) Slice through level set. Blue region indicates interior of object.](image)

However, level sets are difficult to use for registration and tracking tasks because there is no innate ability to track vertices as in the mesh deformation framework. The surface only exists when an iso-surface is extracted from the level set. Furthermore, the level set is stored as an image that is re-sampled at each time step. Re-sampling an image acts as a low-pass filter that results in feature loss as a function of the number of time steps, even if the motion is rigid (e.g. global registration) or divergence free.

### 2.2.5 Particle Systems

An alternative to meshes and level sets is to use particles systems [70-72]. These models use a collection of spherical particles with fixed radii and associated radial basis functions to represent
objects (see Figure 8). The model is deformed by moving particles in response to external forces and inter-particle forces. A scalar function is defined on the particles, which is a linear combination of the particles' radial basis functions. An iso-surface can be extracted from the 3D scalar function in the same procedure as for level sets. The important difference between particles and level sets is that for particles, the scalar function is not signed, so the notion of inside/outside is under-specified. One could choose any iso-level to be the boundary of the model, whereas a level set's zero iso-level always indicates the boundary.

Particle systems are preferred for modeling fluids or other incompressible because the total mass of particles is conserved at each time step. Because particles are a parametric representation (i.e. particles are parameterized by a set of point locations and radii), they can be explicitly tracked and will not lose features under rigid motion. However, particle systems are not good at describing objects with sharp features because the maximal curvature is controlled by the band-width of the radial basis functions, which is dependent on particle radii. Reducing particle radii by 1/2 to increase the maximal curvature by 2 requires 8 times more particles. There are split and merge methods for re-sampling particles to have smaller radii in regions with shaper features [73, 74], but re-sampling will interfere with tracking. Particle tracking is not the same as surface tracking; information associated with the boundary may have no meaning or corollary inside the object (e.g. surface texture mappings). Particle systems could be combined with parametric surface representations (e.g. triangle meshes) to provide both volumetric and surface tracking.

Finally, it is difficult to monitor and control topology of particle systems because the surface is represented implicitly and under specified. The topology of the object can be different depending on the iso-level selected, and digital connectivity rules used to monitor and control the topology of level sets [75] have not been extended to particle systems.
2.2.6 Currents

Currents represent objects as a measurement of flux that passes through a surface. By representing objects in this way, Currents transform surfaces into a linear space that has computable norms [76-78]. This shape space provides a natural theoretical framework in which distances and trajectories between shapes can be discussed. Meshes, level sets, and surfels have Current representations. A useful property of Currents is that they do not require surface elements to be connected. For example, a triangle mesh represented as a Current can be compressed and re-expressed as a constellation of triangles [79]. Springls can be transformed into Currents with the same procedure as triangle meshes, enabling them to be analyzed and compared to other shapes in a linear space regardless of the original geometric representation.

2.2.7 Complete Distance Field Representation (CDFR)

The Complete Distance Field Representation (CDFR) [80] maintains a dual level set and triangle mesh representation of an object. Instead of just interpolating the level set image, CDFR computes the exact distance to the triangle mesh when measuring distances close to the level

\[ \text{http://developer.nvidia.com/opencl-sdk-code-samples} \]
set's zero iso-surface. The benefit of CDFR is that the model's accuracy is not limited by the level set's grid resolution, which is important when rendering objects with sharp edges, corners, or fine detail.

### 2.2.8 Other Representations

There are other deformable model paradigms (see Nealen et al. [81] for a survey of physically-based deformable models). Model selection is motivated by properties of the material being modeled, and there is no representation that is superior for modeling all materials. Different phases of the image analysis pipeline require the model to behave as different types of materials: a rigid solid during registration, elastic solid during non-rigid registration, and plastic or viscoelastic during segmentation. This leads to imaging pipelines with transformations between model representations.

Attempts have been made to unify deformable model representations with varying success through either volumetric [82-86] or surface [48, 87-91] approaches. One of the main challenges these works address is how to re-sample or re-mesh the model during deformation, given their representation. Surface elements are more appropriate for imaging problems because objects are, by definition, “homogeneous” in some sense and differentiated by variations in image intensities that occur near their boundaries. Of these surface-based representations, the Marker Level Set (MarkerLS) [90] and surfels [91] are closest to this work.

**Marker Level Set**

The MarkerLS method [90] maintains a set of particles located on the level set’s zero iso-level. Since particles lie exactly on the zero iso-level, they can be used for tracking the model’s boundary. After each level set and particle advection step, the level set is corrected so that particles continue to lie on the level set’s zero iso-level. Particles are added to cover the zero iso-level and deleted to prevent over-sampling. The MarkerLS method associates a label with each particle to encode tracking information. When a new particle is added, it is assigned a label by interpolating the label of its neighbors.
The main difference between SpringLS and MarkerLS is that SpringLS define the model's level set, whereas MarkerLS uses particles to correct errors in the level set. MarkerLS requires the deformation method to have an equivalent level set and parametric interpretation in order to deform both representations. There are applications in imaging and physical simulation where there is only a parametric interpretation (e.g., Point Distribution Models (PDM) [37], character animation, boundary value problems) for which MarkerLS is not applicable. This thesis will present data structures and methods that can be used for fluid-like deformations and applications where there is only a parametric interpretation.

**Surfels**

Systems for point-sampled shape modeling [91, 92] also relate to this work. In these systems, a manifold surface is approximated by a set of points or disk-shaped surface elements (surfels); from which, a watertight surface can be inferred using Moving Least Squares [93]. The model is deformed by perturbing or stretching surfels. The deformation process creates holes and irregular sampling patterns in the model's representation. To mitigate these problems, several filtering operations are performed. To prevent over-stretching, elongated surfels are split in two; to regularize the sampling distribution, neighboring surfels repel each other; and to fill holes, surfels are inserted. Filtering assumes that the object is smooth, and therefore neighboring surfels can be projected onto a tangent plane with minimal geometric distortion.

Both point-sampled modeling and SpringLS have an auxiliary implicit representation; however, a level set has more utility for imaging applications than an MLS surface because a level set maintains an inside/outside assignment for each voxel in an image at all times during the deformation. This is important for determining / enforcing a model's topology and correcting normals.

Point-sampled modeling and SpringLS differ in their choice of surface element (Figure 9). SpringLS use triangular surface elements instead of disk surface elements to capture all the detail at the resolution of the triangles, whereas point-sampled models define a level set whose sharpness is limited by smoothness assumptions. Methods to approximate triangle meshes or level sets with surfels are challenging [94-96] because they require the introduction of more
elements or a new type of element (i.e. clipped surfels as described by Pauly et al. [91]) to express regions with high curvature. SpringLS can express high curvature regions without approximation.

Figure 9: Igael model represented by a constellation of (a) springls and (b) surfels. Springls were rendered with volumetric raycasting and surfels were rendered with Pointshop3D\(^3\). Insets show constellation of surface elements. The constellation in (a) was re-expressed as surfels to use in the rendering for (b). Consequently, the sampling pattern and number of surface elements is the same.

\(^3\) http://graphics.ethz.ch/pointshop3d/
2.3 Appearance Models

A common approach to solving image segmentation problems is to approximate the forward imaging process based on an estimate of the object's geometry and appearance, compare the synthesized image against the actual image, and use the residual error to guide the model's deformation. Active Appearance Models [38] and Mumford-Shah Models [97] were the earliest works. Since then, forward imaging models have been used for Lambertian surface reconstruction [98], surface reconstruction with unknown illumination [99], Tomographic reconstruction [100], Hybrid Cone-beam Reconstruction [101], and this author's work on reconstruction with C-Arm X-ray images (Chapter 4) [102].

There are solutions for simulating fluid-like materials with triangle meshes that perform online re-meshing and topology changes [47, 103, 104], which begs the question whether level set methods are still needed since their only apparent advantages are in re-meshing. The answer is yes, level set methods are still needed because accurate simulation of the forward imaging process requires ray-tracing. Level sets are easier to ray-trace on graphics hardware than triangle meshes [105, 106], especially if the model is deforming.

Level sets can determine if a location is inside or outside an object in $O(1)$ time (one memory read and one comparison). This allows development of fast ray-tracing methods [107-109], and fast generation of synthetic images based on geometry and appearance models [97, 100, 102], which are important in image segmentation. There are fast localization methods for triangle meshes [110-113], but these require additional data structures that must be recomputed when the mesh undergoes large deformations. The computation time required to rebuild these auxiliary data structures adds significant overhead that is not required for level set methods. Level sets have clear advantages over triangle meshes for applications that require both fluid-like deformations and appearance models. Chapter 3 on multi-view reconstruction and Chapter 7 on image atlas construction are examples of such applications.
2.4 Summary

Table 3 summarizes geometric properties of meshes, level sets, label masks, and Multi-Object Spring Level Sets (MUSCLE). MUSCLE has properties of both meshes and level sets. Note that neither meshes nor level sets are good for modeling sliding surfaces. This is a problem in medical imaging because many structures in the human body slide against each other. The ability to model sliding surfaces is becoming increasingly important as systems are built to model motion of all structures in the human body (e.g., Japan's construction of a 10 Peta FLOP supercomputer for human body simulation). Meshes have difficulty with sliding surfaces because it is easy to create air-gaps and overlaps between adjacent structures as they slide along each other. Level sets do not create air-gaps or overlaps, but the representation must be augmented to model information that moves tangentially along a surface [114, 115]. MUSCLE augment level sets with triangle meshes to preserve properties of both.

Table 3: Comparison of geometric representations. Check marks indicate the preferred representation for each behavior. Multiple check marks indicate those representations exhibit the geometric property equally well.

<table>
<thead>
<tr>
<th></th>
<th>Label Mask</th>
<th>Level Set</th>
<th>Mesh</th>
<th>MUSCLE⁴</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology Change</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Self-intersection Prevention</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collision Handling</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Fluid-like Deformation</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Interacting Fluids</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Simulation of Imaging Process</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Rigid Transform</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Elastic Deformation</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Topology Preservation</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point Tracking</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Active Appearance Models</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Sliding Surfaces</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

⁴ Multi-object Spring Level Sets (MUSCLE) will be presented in Chapter 6.
Chapter 3

3 Reconstruction from C-arm X-ray Images

3.1 Motivation

Re-meshing is the main drawback to modeling highly deformable materials with parametric surface representations. One could argue that re-meshing is not an issue anymore because software packages exist that address the problem reasonably well (e.g. TransforMesh [116]). However, there are image segmentation and reconstruction problems for which level sets are clearly superior for several reasons: surface tracking is not needed, the object’s topology is unknown and likely to change, the object has fluid-like properties, the forward imaging process needs to be accurately simulated with ray-tracing, and the algorithm needs to know which image voxels are inside or outside the object at all times. The following reconstruction method is one such application that motivates the need for level sets.

A novel algorithm is presented to segment and reconstruct injected bone cement from a sparse set of X-ray images acquired at arbitrary poses. The Sparse X-ray Multi-view Active Contour (SxMAC – pronounced “smack”) can (1) reconstruct objects for which the background partially occludes the object in X-ray images, (2) use X-ray images acquired on a non-circular trajectory, and (3) incorporate prior CT information. The algorithm’s inputs are pre-processed X-ray images, their associated pose information, and prior CT, if available. The algorithm initiates automated reconstruction using visual hull computation from a sparse number of X-ray images. It then improves the accuracy of the reconstruction by optimizing a geodesic active contour. Experiments with mathematical phantoms demonstrate improvements over a conventional silhouette based approach, and a cadaver experiment demonstrates SxMAC’s ability to reconstruct high contrast bone cement that has been injected into a femur and achieve sub-millimeter accuracy with 4 images.
3.2 Background

The problem of recovering 3D shape from a sparse set of 2D projection images is common in interventional imaging. If prior information such as a statistical shape model is available, this information may be used to assist in reconstruction [117]. However, such information is not always available, especially if the object is highly deformable or its shape is created and/or substantially modified during the procedure. Examples include surgical procedures for injecting cement into bones, such as vertebroplasty [118], sacroplasty [119], and femoroplasty [120].

Our immediate, focusing clinical application is femoral bone augmentation, in which cement is injected into the neck and intertrochanteric area of the femur for patients with severe osteoporosis. The goal is to strengthen the femur to prevent fractures [120]. In a recently proposed procedure for bone augmentation (see Figure 10) [121], pre-operative CT images are used for pre-operative planning, based on a 3D finite element analysis of the patient’s femur and planned cement injection [122]. The pre-operative model and plan are registered to the patient and intra-operative navigation system using 2D-3D registration from intra-operative X-rays, and a robotic device is used to inject cement containing an appropriate contrast agent. At various stages during the injection, a sparse set of intra-operative X-rays (at most 8, but preferably 4) are taken of the cement and the cement volume in the bone is estimated. This information is used to repeat the finite-element analysis of augmented bone strength and to support re-planning and optimization for further injections. Conventionally, the shape of the cement volume is estimated by intersecting cones formed from the silhouettes of the cement in the images. However, the resulting models do not accurately reflect the actual cement volumes (e.g. Figure 12b as compared to ground truth shown in Figure 12a). Our goal in this work is to significantly improve the accuracy of this reconstruction while still using only a small number of intra-operative X-rays from a conventional C-arm.
Techniques have been developed in computer vision to reconstruct objects observed from multiple viewpoints without prior information. One classical approach is to segment an object’s silhouette in images, back-project the silhouettes into 3D space, and compute the intersecting volume. This technique is known as silhouette reconstruction or visual hull computation, and has been used in computer vision and C-arm X-ray reconstruction. It has been shown that the visual hull encloses all other reconstructions that can be explained by 2D segmentations of an object. The visual hull is unlikely to be consistent with observed image intensities of the object. However, the visual hull can be used to initialize more sophisticated approaches that generate reconstructions consistent with image intensities. In particular, Geodesic Active Contours reconstruct objects by optimizing an objective function on the image intensities that considers all observations of an object simultaneously.

Active contour techniques have been extended to 2D tomographic reconstruction and Cone-Beam CT (CBCT). CBCT is a volumetric reconstruction from a series of X-ray images (typically more than 100) acquired with a 2D X-ray image scanner, such as a C-arm. The active contour segmentation process is formulated as the minimization of the following objective function

$$E = E_{model}(X) + E_{data}(\mu(X), I_1(x_1), \ldots, I_K(x_K)).$$  (3.1)
with respect to the linear attenuation coefficients \( \{\mu(X) | X \in D \subset \mathbb{R}^3\} \) given \( K \) X-ray images \( \{l_k(x_k) | x_k \in D[k] \subset \mathbb{R}^2\} \) in log space. \( x_k \)'s refer to coordinates in each 2D X-ray image. The objective function also incorporates information about geometric properties of the object \( (E_{model}) \), such as a penalty against high curvature. See Figure 11 for a depiction of the imaging scenario and Table 4 for definitions of all major terms.

Figure 11: Imaging scenario depicting (a) X-ray images, (b) X-ray source, (c) projection lines, (d) background objects, and (e) foreground object.
Table 4: Definition of terms and expressions.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k \in D[k] \subset \mathbb{R}^2$</td>
<td>Coordinates in the domain of a 2D X-ray image $k$.</td>
</tr>
<tr>
<td>$X \in D \subset \mathbb{R}^3$</td>
<td>Patient coordinates in 3D.</td>
</tr>
<tr>
<td>$X_n \in \mathbb{R}^3$</td>
<td>Coordinates of the $n^{th}$ voxel in a 3D image.</td>
</tr>
<tr>
<td>$I_n \in \mathbb{R}^{1 \times N}$</td>
<td>Indicator vector for which the $n^{th}$ column is 1 and all other columns</td>
</tr>
<tr>
<td></td>
<td>are 0.</td>
</tr>
<tr>
<td>$P_{fg}(x_k, \mu)$</td>
<td>Probability distributions of linear attenuation coefficients in X-ray</td>
</tr>
<tr>
<td></td>
<td>image $k$ for pixel location $x_k$.</td>
</tr>
<tr>
<td>$P_{bg}(x_k, \mu)$</td>
<td>Probability distributions of linear attenuation coefficients in X-ray</td>
</tr>
<tr>
<td></td>
<td>image $k$ for pixel location $x_k$.</td>
</tr>
<tr>
<td>$D_{fg}[k]$ and $D_{bg}[k]$</td>
<td>Foreground (fg) and background (bg) domains in X-ray image $k$.</td>
</tr>
<tr>
<td>$\mu_{fg}(\cdot)$ and $\mu_{bg}(\cdot)$</td>
<td>Linear attenuation coefficients for foreground and background regions</td>
</tr>
<tr>
<td></td>
<td>in 3D.</td>
</tr>
<tr>
<td>$I_k(\cdot)$</td>
<td>The log X-ray intensity image.</td>
</tr>
<tr>
<td>$H(\cdot)$</td>
<td>Heaviside function.</td>
</tr>
<tr>
<td>$DRR_k(\mu)(x)$</td>
<td>Simulated X-ray image $k$ generated from volume $\mu$.</td>
</tr>
<tr>
<td>$\text{vec}(M)$</td>
<td>A column vector representing the elements of matrix $M$.</td>
</tr>
<tr>
<td>$\delta(x) = \frac{d}{dx}H(x)$</td>
<td>Dirac delta corresponding to the Heaviside function $H(\cdot)$.</td>
</tr>
<tr>
<td>$p_k(X)$</td>
<td>Mapping from patient coordinate space to pixel coordinates $c$.</td>
</tr>
<tr>
<td>$M_k \in \mathbb{R}^{M \times N}$</td>
<td>System matrix approximating the X-ray imaging equation.</td>
</tr>
<tr>
<td>$\omega_{fg}(X)$ and $\omega_{bg}(X)$</td>
<td>3D Level set corresponding to the foreground (fg) and background (bg)</td>
</tr>
<tr>
<td></td>
<td>object. Level sets are negative inside the object and positive outside.</td>
</tr>
<tr>
<td>$\psi_k(x_k)$</td>
<td>Silhouette’s level set in image $k$.</td>
</tr>
<tr>
<td>$\lambda_k$</td>
<td>Curvature weight.</td>
</tr>
<tr>
<td>$H_{fg}(X)$ and $H_{bg}(X)$</td>
<td>Indicator functions for foreground and background.</td>
</tr>
</tbody>
</table>

The objective function (eq. 1) is optimized by alternating between minimization of $E_{\text{model}}$ and $E_{\text{data}}$ until convergence. One choice for the data term uses the log-likelihood estimator on the foreground $P_{fg}(x_k)$ and background $P_{bg}(x_k)$ probability densities [133],

$$E_{\text{data}} = \sum_{k} \int_{D_{fg}[k]} \left( -\log \left( P_{fg}(x_k) \right) \right) dx_k + \int_{D_{bg}[k]} \left( -\log \left( P_{bg}(x_k) \right) \right) dx_k, \quad (3.2)$$

where the foreground $D_{fg}[k]$ and background $D_{bg}[k]$ are assumed to appear as disjoint regions in X-ray images. The objective function can be expressed using a Heaviside function $H(\cdot)$ on the silhouette’s level set $\{ \psi_k(x_k) | x_k \in D[k] \subset \mathbb{R}^2 \}$ such that the level set is positive outside and negative inside the silhouette of the deformable model:
\[ E_{\text{data}} = - \sum_{k}^{K} \int_{D(k)} \log \left( P_{fg}(x_k) \right) \left( 1 - H(\psi_k(x_k)) \right) + \log \left( P_{bg}(x_k) \right) H(\psi_k(x_k)) \, dx_k. \quad (3.3) \]

There may not exist prior information about the probability distributions of the X-ray intensities. Instead, the active contour may use edge information \[133\] to find object boundaries or a Mumford-Shah approach. Alvino and Yezzi \[130\] proposed the following Mumford-Shah objective function for CT reconstruction:

\[
E_{\text{data}} = \sum_{k}^{K} \int_{D} \left( (I(s, \theta_k) - f(s, \theta_k))^2 ds 
\right.

\left. + \int_{D} \left\{ \lambda_{fg} \left| \nabla \mu_{fg}(x) \right|^2 H_{fg}(x) + \lambda_{bg} \left| \nabla \mu_{bg}(x) \right|^2 H_{bg}(x) \right\} \, dx \right) \quad (3.4)
\]

where,

\[
f(s, \theta) = \int (\mu_{fg}(x) H_{fg}(x) + \mu_{bg}(x) H_{bg}(x)) \delta(x \cos \theta + y \sin \theta - s) \, dx. \quad (3.5)
\]

\(f(s, \theta)\) is the radon transform of \(\mu(X)\). The latter Mumford-Shah approach has the advantage that it does not require prior knowledge of the foreground and background appearance. However, it assumes the foreground and background densities are smoothly varying, which may not be the case. To achieve high accuracy, the objective function must be representative of the anatomical contents and imaging scenario for a specific application.

To serve the needs of the bone augmentation procedure previously described, we focus on reconstructing homogenous highly deformable objects (i.e. bone cement) from C-arm X-ray projection images. In this paper, the segmentation process is formulated as an optimization problem that permits the segmentation algorithm to (1) reconstruct deformable objects for which the background partially occludes the object in X-ray images, (2) use X-ray images acquired on a non-circular trajectory, and (3) incorporate prior CT information. Subsequently, we describe a method for optimizing the objective function and evaluate the feasibility and
performance of the Sparse X-ray Multi-view Active Contour algorithm (SxMAC) to reconstruct injected bone cement. In particular, we are interested in knowing how many X-ray images, how much contrast, and how large a sweep angle is required to achieve acceptable accuracy.

### 3.3 Method

Following the approach by Alvino and Yezzi [130], the segmentation process is formulated as an optimization problem that minimizes the following objective function, which describes the disparity between X-ray images and Digitally Reconstructed Radiographs (DRRs) of a deformable model:

$$
E_{data} = \sum_k^{K} \int_{D[k]} (I_k(x_k) - DRR_k(\mu)(x_k))^2 dx_k
+ \int_D \left\{ \lambda_{fg} |\nabla \mu_{fg}(X)|^2 H_{fg}(X) + \lambda_{bg} |\nabla \mu_{bg}(X)|^2 H_{bg}(X) \right\} dX
$$

where,

$$
DRR_k(\mu)(x_k) = \int_D \mu(X) \delta(x_k - p_k(X)) dX,
$$

$$
\mu(X) = \mu_{fg}(X)H_{fg}(X) + \mu_{bg}(X)H_{bg}(X),
$$

$$
H_{fg}(X) = \left(1 - H(\omega_{fg}(X))\right)\left(1 - H(\omega_{fg}(X))\right),
$$

and,

$$
H_{bg}(X) = \left(1 - H(\omega_{bg}(X))\right)H(\omega_{fg}(X)).
$$

The objective function $E_{data}$ measures the $L^2$ norm of the difference between simulated X-rays of $\mu$ and the log of each X-ray image, subject to an $L^2$ penalty on the smoothness of $\mu$. 

31
Alternatively, the objective function could be expressed with an $L^1$ instead of $L^2$ norm, for which there is evidence that the $L^1$ norm may have better performance if there exist a sparse representation of image intensity information [134, 135]. Likewise, the smoothness term (second term in eq. 6) could minimize the Total Variation of the foreground and background appearance $|\nabla \mu(X)|$. The following discourse will focus on $L^2$ optimization because the objective function can be efficiently solved with linear methods. The complete objective function for the $L^2$ norm, including data and geometric terms, is as follows,

$$E = \sum_{k} \int_{D[k]} (I_k(x_k) - DRR_k(\mu)(x_k))^2 \, dx_k$$

$$+ \int_{D} \left( \lambda_{fg} |\nabla \mu_{fg}(X)|^2 H_{fg}(X) + \lambda_{bg} |\nabla \mu_{bg}(X)|^2 H_{bg}(X) \right) \, dX.$$  

(3.11)

$DRR_k(\cdot)$ can be discretized and expressed as a weighted linear combination of $\mu(X)$:

$$DRR_k(\mu)(x_k) = \sum_{\{X|X_k=p_k(X)\}} w_k(X) \mu(X)$$  

(3.12)

or alternatively,

$$\text{vec}(DRR_k(\mu)) = M_k \text{vec}(\mu).$$  

(3.13)

$M_k$ is an $M \times N$ matrix where $M$ is the number of pixels in the X-ray image and $N$ is the number of voxels in $\mu$. The matrix is completely defined by X-ray geometry (extrinsic and intrinsic parameters) and does not depend on the image or volume intensities. To efficiently solve eq. 11, we let the foreground and background appearances be modeled as a constant (i.e. $\mu_{fg}(X) = \mu_{bg}(X) = \mu_0$)
\( c_{fg} \) and \( \mu_{bg}(X) = c_{bg} \). This assumption is consistent with work by Chan and Vese [136]. The objective function simplifies to,

\[
E = \sum_{k} \left\| \text{vec}(I_k) - c_{fg} M_k \text{vec}(H_{fg}) - c_{bg} M_k \text{vec}(H_{bg}) \right\|^2 + \int_{D} \lambda_{k} \delta \left( \omega_{fg}(X) \right) \left| \nabla \omega_{fg}(X) \right| \, dX
\]  

(3.14)

where \( M_k \) is the system matrix approximating the X-ray imaging equation. The SxMAC model can be augmented to incorporate prior CT information by replacing the background indicator \( H_{bg}(X) \), with the prior CT \( (\mu_{\text{CT}}) \) :

\[
\tilde{H}_{bg}(X) = H_{bg}(X) \mu_{\text{CT}}(X).
\]  

(3.15)

This extension assumes \( \mu_{\text{CT}} \) is properly registered and intensity calibrated so that the background DRR is highly correlated with the background observed in acquired X-ray images. \( H_{bg}(X) \) is a segmentation mask of the field of view that is common to both the prior CT and C-arm acquisition that encloses the foreground \( H_{fg}(X) \).

Alternatively, we can incorporate prior information by replacing \( I_k \) with the difference \( \Delta I_k \) between X-ray images obtained before and after cement injection. Either way, the optimization procedure is the same.

Solving the Euler-Lagrange equation for \( c_{fg} \) and \( c_{bg} \),

\[
0 = \frac{\partial E}{\partial c_{fg}} \bigg|_{c_{fg}} = -2c_{fg} \sum_{k} \text{vec}^T(H_{fg}) M_k \text{vec}(I_k) - M_k \text{vec}(\mu).
\]  

(3.16)

From eq. 16 we obtain,

\[
\hat{c}_{fg}(c_{bg}) = \sum_{k} \left( M_k \text{vec}(H_{fg}) \right)^T \left( \text{vec}(I_k) - c_{bg} M_k H_{bg} \right) \\
\left\| M_k \text{vec}(H_{fg}) \right\|^2
\]  

(3.17)
and similarly,

\[
\hat{c}_{bg}(c_{fg}) = \sum_k^K \frac{(M_k \text{vec}(H_{bg}))^T \left( \text{vec}(l_k) - c_{fg} M_k(H_{fg}) \right)}{\| M_k \text{vec}(H_{bg}) \|^2}
\]  

(3.18)

where \( M_k^T \) is the adjunct operator approximating the back-projection of the X-ray image into 3D space. \( \hat{c}_{fg} \) and \( \hat{c}_{bg} \) are new estimates for the foreground and background intensities based on their current estimates \( (c_{fg} \text{ and } c_{bg}) \). The model’s appearance can be optimized by alternating between \( c_{fg} = \hat{c}_{fg}(c_{bg}) \) and \( c_{bg} = \hat{c}_{bg}(c_{fg}) \). Evolution of the deformable model’s level set \( \omega_{fg}(X) \) is computed by gradient descent [127],

\[
\omega_{fg}^{t+1}(X) = \omega_{fg}^t(X) - \gamma \frac{\partial E}{\partial \omega_{fg}^t}
\]  

(3.19)

where,

\[
\frac{\partial E}{\partial \omega_{fg}^t} \bigg|_{X=X_n} = 2(c_{fg} - c_{bg}) \delta \left( \omega_{fg}(X_n) \right) \sum_k^K \| n \| M_k^T \left( \text{vec}(l_k) - M_k \text{vec}(|\mu|) \right) +
\]

\[
\lambda_k \delta \left( \omega_{fg}(X_n) \right) \nabla \cdot \frac{\nabla \omega_{fg}(X_n)}{\| \nabla \omega_{fg}(X_n) \|}.
\]

(3.20)

\( M_k \) and \( M_k^T \) are referred to as forward and backward projection operators, respectively. In the actual implementation, we do not store \( M_k \) and \( M_k^T \) because they are very large sparse matrices. Instead, the graphics card is used to compute elements of those matrices on-the-fly with OpenCL. We choose a voxel driven approach for its simplicity [137, 138], but more accurate methods have been developed that can be implemented on the graphics card [139-141]. The SxMAC reconstruction algorithm proceeds as follows:
Algorithm 1. SxMAC

<table>
<thead>
<tr>
<th>foreach ( n = 1: N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>// Optimize appearance</td>
</tr>
<tr>
<td>do</td>
</tr>
<tr>
<td>( \hat{c}<em>{fg}(c</em>{bg}) \rightarrow c_{fg} )</td>
</tr>
<tr>
<td>( \hat{c}<em>{bg}(c</em>{fg}) \rightarrow c_{bg} )</td>
</tr>
<tr>
<td>while ( \hat{c}<em>{fg}(c</em>{bg}) \neq c_{fg} ) and ( \hat{c}<em>{bg}(c</em>{fg}) \neq c_{bg} )</td>
</tr>
<tr>
<td>// Evolve level set</td>
</tr>
<tr>
<td>do</td>
</tr>
<tr>
<td>( \omega_{fg}^{t+1}(X) = \omega_{fg}^{t}(X) - \gamma \frac{\partial E}{\partial \omega_{fg}} )</td>
</tr>
<tr>
<td>while ( \exists X ) s. t. (</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

The Euler-Lagrange optimization procedure is only guaranteed to find local optima. Therefore, it’s important to initialize the model close to the globally optimal solution. When possible, we initialize SxMAC with the object’s visual hull obtained from automated silhouette reconstruction.

3.4 Results

3.4.1 Phantom Experiments

SxMAC was evaluated on mathematical phantoms of solid objects. Synthetic X-ray projections for the objects were generated with the same DRR operator used in the segmentation process. Table 5 lists parameters used in phantom experiments that reflect realistic C-arm X-ray imaging parameters. Figure 12 depicts reconstruction of a metasphere from 2 X-ray images spaced 90° apart \( (\lambda_\kappa = 0.05) \). SxMAC recovers concavities that are not recoverable with silhouette reconstruction alone. Figure 12 depicts a torus reconstructed from 4 images \( (\lambda_\kappa = 0.1) \) with different angular spacing. Notice that the deformable model better captured the hole in the middle of the torus when the acquisition trajectory was ±10° above and below the orbital plane. Figure 14 shows a reconstruction of a dragon from 6 images \( (\lambda_\kappa = 0.01) \) acquired from a circular trajectory. The regularization parameter \( \lambda_\kappa \) should be chosen large enough to reduce
directional bias, but small enough to preserve sharp object features. The dragon reconstruction demonstrates SxMAC has applications in the broader realm of 3D model acquisition.

Figure 12: (a) Metasphere phantom with (b) corresponding silhouette, (c) SxMAC reconstruction, and (d,e) DRRs (eq. 12).
Figure 13: SxMAC reconstruction of a torus acquired from a (a,e) 90° arc, (b,f) 30° arc, and wobbled (c,g) 30° arc. X-ray projection images are depicted in their proper pose relative to the reconstructed object. Reconstructions shown in (a), (b), and (c) are juxtaposed to (d) ground truth in (e), (f), and (g).
Figure 14: (a) Dragon phantom ground truth, (b) visual hull initialization, (c) SxMAC reconstruction, and (d) X-ray images shown in their relative pose.

Table 5: Parameters for Phantom Experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Detector Distance</td>
<td>900 mm</td>
</tr>
<tr>
<td>Detector Dimensions</td>
<td>288 mm x 288 mm</td>
</tr>
<tr>
<td>Pixel Size</td>
<td>0.45 mm</td>
</tr>
<tr>
<td>( \lambda_{fg} )</td>
<td>1</td>
</tr>
<tr>
<td>( \lambda_{bg} )</td>
<td>1</td>
</tr>
<tr>
<td>Source Detector Distance</td>
<td>900 mm</td>
</tr>
<tr>
<td>Detector Dimensions</td>
<td>288 mm x 288 mm</td>
</tr>
</tbody>
</table>

3.4.2 SxMAC Reconstruction with Synthesized Images

Cadaver experiments were conducted to evaluate a surgical procedure in which bone cement is injected into an osteoporotic femur. One objective of the procedure is to provide feedback to the surgeon via intra-operative reconstruction of the injected bone cement. Predictions about the post-operative structural properties of the femur can be made from its 3D reconstruction
and pre-operative CT [122]. The imaging scenario has been described in previous work [121] as follows:

1. Either a pre-operative diagnostic CT or CBCT is acquired.
2. C-arm X-ray images are acquired and registered to the pre-operative CT with 2D/3D registration.
3. Pre-injection X-ray images are taken to guide the procedure.
4. Bone cement is injected into the femur and X-ray images are acquired with the C-arm.
5. SxMAC is used to reconstruct the injected bone cement.
6. Finite Element Analysis of the cement is conducted.
7. The injection procedure is re-planned and repeated from step 4 until the surgeon is satisfied with the result.

This workflow is applicable to using either pre-operative CT or pre-injection X-ray images as priors. SxMAC does not address how to register pre-operative CT to intra-operative X-ray images or how to calibrate intensities between DRRs and X-ray images, although solutions have been described in literature [142, 143]. We'll later describe how registration and intensity calibration can be avoided by using pre-injection X-ray images as priors instead of pre-operative CT.

To evaluate the performance of just the reconstruction algorithm, we synthesized X-ray images from pre and post-operative CBCTs (Figure 15). An intra-operative CBCT would not be acquired in a real clinical scenario, which is why a sparse X-ray reconstruction method like SxMAC is necessary. The following pre-processing steps synthesize pre and post-operative CBCT and X-ray images that are perfectly registered and intensity matched:

1. Pre and post-operative CBCTs are acquired with a flat panel C-arm (Table 6). Geometric calibration follows the method described by Daly, Siewerdsen et al. [144].
2. Pre-operative CBCT is registered to post-operative CBCT using intensity-based 3D/3D registration.
3. The femur is segmented in pre-operative CBCT.
4. The cement is segmented in post-operative CBCT and the Volume of Interest (VOI) is copied and pasted into the registered pre-operative CBCT.
5. DRRs for the CBCTs with and without the bone cement are generated.
Figure 15: Synthesized (a) pre-operative X-ray image of dry femur and (b) post-operative X-ray image with cement attenuation of 1900 HU.

Table 6: Flat panel C-Arm specification.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>Siemens Medical Solutions</td>
</tr>
<tr>
<td>Type</td>
<td>Mobile isocentric flat-panel cone-beam CT</td>
</tr>
<tr>
<td>Scan time</td>
<td>128 sec.</td>
</tr>
<tr>
<td>Voltage</td>
<td>120 kVp</td>
</tr>
<tr>
<td>Current</td>
<td>5.2 mAs</td>
</tr>
<tr>
<td>Pixel size</td>
<td>0.388 mm x 0.388 mm</td>
</tr>
</tbody>
</table>

SxMAC was evaluated on these synthetic X-ray images. The CBCT and deformable model’s level set representation were sampled at 1 mm isotropic, and the regularization parameter was chosen to be $\lambda_c = 0.001$. The background $\omega_{bg}(X)$ corresponded to a segmentation of the femur. SxMAC was initialized with a sphere (10 mm radius) located at the center of the bone cement. Such initialization is clinically plausible because the cement injector’s tip is tracked with a Polaris optical tracker as part of the bone augmentation procedure. We were unable to reliably initialize SxMAC with silhouette reconstruction because it failed in several experiments due to lack of contrast.

In the first set of experiments, the attenuation of the cement was varied from 1645 HU to 2070 HU at equal intervals, and the number of images was held constant at 8 images (Figure 16). In the second set of experiments, the number of images was varied from 2 to 10, and the cement’s
attenuation was held constant at 1900 HU (Figure 17). Each collection of images was evenly sampled in a plane from a 180° arc trajectory. In the third set of experiments, the sweep angle was varied between 20° and 180° (Figure 18), and the number of images and cement attenuation were held constant at 8 and 1900 HU, respectively.

**Figure 16:** Error between ground truth and SxMAC reconstruction for varying bone cement attenuation. Error bars indicate one standard deviation from the mean. Error was measured as the average point-to-surface distance for either mesh vertices on the SxMAC reconstruction to ground truth surface (error relative to reconstruction) or vice versa (error relative to ground truth).

**Figure 17:** Error between ground truth and SxMAC reconstruction for different numbers of synthetic X-ray images.

In the final set of experiments, the pre-operative CBCT was misaligned with the post-operative CBCT, either by translating it in a random direction or rotating it around a random axis, to simulate the effect of registration error. Experiments measuring the accuracy of SxMAC
reconstruction were repeated 10 times per translation (Figure 19) and rotation (Figure 20). We compensate for 2D/3D registration error with 2D Affine registration of pre- and post-operative X-ray images, also plotted in Figure 19 and Figure 20.

Intra-operative reconstruction errors were measured in terms of distance relative to mesh vertices on the ground truth reconstructed surface (truth-to-reconstruction) or vice versa (reconstruction-to-truth). A 180° sweep angle was required to obtain sub-millimeter accuracy of 1.0±0.75 (4 images) and 0.85±0.62 (8 images) mm for reconstruction-to-truth and 0.84±0.52 (4 images) and 0.73±0.51 (8 images) for truth-to-reconstruction. Reconstructions are depicted in Figure 21. SxMAC appears to improve the accuracy of cement reconstruction (Figure 21c,d) over silhouette reconstruction (Figure 21b). Silhouette reconstruction tends to underestimate the true volume because edges around the cement appear faint in X-Ray images (Figure 15b). SxMAC reconstruction is sensitive to image contrast, which is dependent on the cement attenuation.

![Figure 18: Error between ground truth and SxMAC reconstruction from 8 images for different sweep angles.](image-url)
Figure 19: Error between ground truth and SxMAC reconstruction from 4 images and different magnitude translation errors.

Figure 20: Error between ground truth and SxMAC reconstruction from 4 images and different magnitude rotation errors.
Results demonstrate SxMAC’s ability to use intensity information to recover shape information that is not recoverable with silhouette reconstruction alone. Results from the cadaver experiment demonstrate SxMAC’s reconstruction performance with prior CT. Figure 16 suggests that SxMAC reconstruction can be greatly improved by increasing the bone cement attenuation from 1550 HU to at least 1900 HU. An increase in attenuation could be achieved by using a higher concentration of Barium in the PMMA bone cement. Figure 17 and Figure 18 demonstrate sub-millimeter accuracy is achievable with a minimum of 4 images and 180° sweep angle. Reconstruction accuracy improves with increasing sweep angle because there is less redundant information between views. This trend is evident in phantom experiments (Figure 12). There was a 15% improvement between 4 and 8 images, and only a 1% improvement between 8 and 10 images. A 180° sweep angle is necessary to obtain sub-millimeter reconstruction-to-truth accuracy with either 4 or 8 images. Acquisition of 4 to 8 images is clinically relevant for intra-operative procedures.

Algorithms for 2D/3D registration are reported to have less than 2 mm translation and 1° rotation error in the pelvic region [121, 145, 146]. With the addition of 2D image registration, SxMAC achieves 1 mm accuracy in the range of these misalignment errors and is robust to even larger displacement errors (see Figs. 10 and 11).

### 3.4.3 SxMAC Reconstruction with Real Images

The first cadaver study produced many important findings; however, the image contrast between cement (1550 HU) and surrounding tissue was insufficient to use X-ray images acquired
from that experiment for reconstruction. Observing that reconstruction accuracy is sensitive to the amount of image content that can be explained with prior information, we changed the imaging scenario and conducted a second cadaver study that uses pre-injection X-ray images as priors. This modification circumvents issues with synthesizing realistic X-ray images from pre-operative CT.

In the second study, X-ray images were acquired immediately before and after cement injection from the same poses. With modern C-arms, it is trivial to retake images from previous poses because the C-arm is motorized and position encoded. Care was taken not to move the cadaver between acquisition of pre and post injection images, although small patient motions could be corrected with 2D/2D registration of X-ray images. We then computed the log of the pre and post injection images and subtracted them to produce images that contained only the cement (Figure 22). Additionally, we chose a cadaver with a hip implant in the other leg. This reduced the sweep angle in which the cement was visible to 130°. Occlusion is anticipated because the bone augmentation procedure is expected to occur in conjunction with hip replacement or revision surgery.

![Figure 22: (a) Post-injection X-ray image and (b) Pre/Post-injection difference image.](image)

The cement was first automatically segmented with 2D active contours [136]. A silhouette reconstruction was computed from 2D segmentations and used to initialize SxMAC. To capture spiculated cement features visible in the second experiment, the level set resolution was increased to 0.6 mm isotropic. The regularization parameter ($\lambda_s = 0.05$) was chosen roughly to
match the feature size of the ground truth segmentation and held constant in all experiments. SxMAC was run on collections of images evenly sampled from a 130° arc. Figure 24 and Figure 25 show that sub-millimeter accuracy is achievable with as few as 4 images (0.69±1.03 mm). We then held the number of images constant at 4 and varied the sweep angle for SxMAC reconstruction. Figure 25 shows that 1.54±1.31 mm accuracy is achievable with at least a 90° sweep angle.

Figure 23: (a) Ground truth cement segmentation, (b) silhouette reconstruction from 4 images, (c) silhouette reconstruction from 8 images, (d) SxMAC reconstruction from 4 images, (e) SxMAC reconstruction from 8 images. Red arrow shows sharp edge that is an artifact of silhouette reconstruction.

Figure 24: Error between ground truth and SxMAC reconstruction from for different numbers of real X-ray images. Error was measured as the average point-to-surface distance for either mesh vertices on the SxMAC reconstruction to ground truth surface (error relative to reconstruction) or vice versa (error relative to ground truth).
3.5 Discussion

It is interesting to observe that reconstruction accuracy is better with fewer images. This is because increasing the number of images increases the risk of a poor 2D segmentation in at least one image, leading to less accuracy in silhouette and subsequent SxMAC reconstruction. Consider a pathological case where the cement is occluded in only one of many images. Since the object has no silhouette in that one image, the visual hull reconstruction will contain nothing. Therefore, it is important that the cement is not occluded in any images used for reconstruction. For a thorough discussion of visual hulls, see Laruentini [123].

It may seem surprising that compared to SxMAC, silhouette reconstruction is slightly more accurate when measuring error from truth-to-reconstruction as a function of the number of images in Figure 24b. However, Figure 23d,e shows that SxMAC reconstructions better resemble ground truth and are less biased towards particular imaging directions (notice the sharp edges indicated by red arrows in Figure 23b,c that are an artifact of silhouette reconstruction). The regularization term ($\lambda_w$) is responsible for reducing directional bias and other aliasing artifacts.

SxMAC makes several assumptions that are particular to bone cement injection procedures. Namely, the object is assumed to be homogeneous, located in a particular region of space, and
the image background resembles DRRs of a pre-operative CT or pre-injection image. These assumptions enable SxMAC to produce accurate reconstructions where there is soft tissue, truncation, and metal that would pose challenges for tomographic reconstruction algorithms [147, 148], which operate under fewer assumptions and generally require more images than the 4 needed by SxMAC. Another important difference between SxMAC and cone-beam reconstruction approaches is that SxMAC produces a geometric representation of the object, whereas cone-beam reconstruction produces images that must still be segmented to extract the object’s geometry. Segmenting objects in CBCT images can be challenging because of metal and streak artifacts [149], which are to be expected in intra-operative imaging scenarios.

The synthesized X-ray images used to evaluate SxMAC are representative of ideal results achievable with the following pre-processing pipeline: register the diagnostic CT to X-ray images, generate DRRs from diagnostic CT, and remap the X-ray intensities to DRR intensities. SxMAC’s true performance during an intra-operative procedure will be affected by the performance of these pre-processing steps.

To circumvent challenges posed by 2D/3D registration, we have presented an alternative imaging scenario that uses image subtraction instead of registration. SxMAC’s performance in this scenario is superior to the previous, and the requirements for bone cement contrast and sweep angle are less stringent. The procedure is independent of patient anatomy and thus adaptable to other injection regions, such as the spine. A standard CBCT with our C-arm requires 400 X-ray images. SxMAC achieves sub-millimeter accuracy with 4 images, or 1% of the dosage. This suggests SxMAC can be incorporated into a surgical procedure without affecting patient safety. Moreover, the dosage is low enough that more than one reconstruction could be performed during the procedure to provide more feedback to the surgeon.

Although the intended application for SxMAC is to intra-operatively reconstruct bone cement, there are other surgical procedures that involve bone cement injection, such as vertebroplasty [118], sacroplasty [119], and femoroplasty [120], that could benefit from intra-operative reconstruction. In the same spirit as Varshney et al. [133], SxMAC could also be applied to metal implant reconstruction.
The dragon reconstruction suggests SxMAC has other imaging applications in 3D model acquisition. The advantage of using a C-arm X-ray machine to acquire a 3D model is that X-rays can image occluded regions that could not be imaged with a laser range scanner, structured light system, multi-view stereo system, or similar technology. C-arm X-ray machines are popular at medical institutions, making 3D model acquisition with a C-arm X-ray machine potentially cheaper and more accessible than other technologies.

3.6 Conclusion

This paper has presented an algorithm for 3D segmentation of bone cement observed in a small number of X-ray images. The SxMAC algorithm provides a computationally efficient procedure for segmentation that can incorporate prior CT and an initial estimate for the object’s shape and location. The algorithm is implemented within an automated pipeline whose inputs are pre-processed X-rays, their associated pose information, and prior CT, if available. Cadaver experiments demonstrate SxMAC can segment injected bone cement with sub-millimeter accuracy from as few as 4 X-ray images, and thus it is readily applicable to intra-operative surgical procedures.

The regularization parameter $\lambda_K$ was held constant for each cadaver experiment, but it is an important parameter that should be optimized in the future once the number of images, sweep angle, and cement viscosity are decided upon. Future work will also examine the accuracy and sensitivity of finite element analysis to SxMAC reconstruction of the cement.

**Contributions:**

- New reconstruction algorithm for C-Arm X-Ray machines that requires only 4 to 8 X-ray images.
- Fully automated system that can reconstruct bone cement with sub-millimeter accuracy.
Chapter 4

4 Multi-Object Geodesic Active Contours

4.1 Motivation

Our objective is to unify the three popular geometric data structures in image analysis. We begin by merging level sets and label masks into one representation. The major advantage of level sets is that they have sub-voxel precision, which allows piecewise-linear surfaces to be represented. The drawback of level sets is that they are more difficult to use than label masks for modeling multiple objects. For instance, a classical approach would be to store one level set for each object in an image. However, it would quickly become intractable to store and manipulate hundreds of objects with this representation scheme, whereas the amount of memory needed to store multiple objects with label masks is independent of the number of objects. Multi-Object Geodesic Active Contours (MOGAC) close the performance gap between level sets and label masks.

4.2 Background

The Level Set Method (LSM) [66, 127] is popular in image analysis for segmenting images [150]. LSMs solve PDEs to produce image segmentations with sub-pixel accuracy. The multi-object version is capable of segmenting adjacent structures without gaps or overlaps [68, 69, 151-154]. However, current implementations of the Multi-object Level Set Method (MLSM) are slow and require a large memory footprint compared to their region growing [27] and graph cut counterparts [28], which lack sub-voxel accuracy. A modern challenge is to develop MLSM implementations that have competitive computational and memory efficiency with region growing and graph cut methods.

Several methods have been proposed for segmenting \( L \) objects with \( L \) level set functions [69, 151, 153] that are stored as images. Storage of these level set images is intractable for tasks
such as cell tracking in microscopy images [155] where there are potentially hundreds to thousands of objects. The Multi-phase LSM [68] reduces the number of level sets to $\log_2(L)$, and the Multi-compartment LSM [152, 154] 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 $L$ objects, and the computation time can be independent of the number of objects.

All previous MLSMs use serial implementations of the narrow-band method [66], which requires periodic re-initialization of the signed distance field. One notable exception is the work by Lie et al. [156], but their method does not have sub-pixel precision. The fast-marching method [157] for distance field re-initialization is a computational bottleneck whose complexity is $O(M^d \log M)$ for $M^d$ pixels in dimension $d \in \{2,3\}$. A more efficient approach is to use the sparse-field method [158] 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, interactive, implementations must leverage parallelism now abundant on modern GPUs and CPUs. There has been work on parallel implementations of single object LSMS [159-161] 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.

4.3 Overview

The contributions of this work are two-fold. First, we describe how to represent $L$ level sets with only a label mask and distance field. Second, we describe a parallel algorithm for segmenting $L$ objects in 2D and 3D. Properties of the sparse-field LSM are leveraged to represent and evolve
$N$ level sets with the "label mask + distance field" data structure. 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. We regard parallelism as a necessary consideration when proposing a new algorithm because computing hardware is becoming more parallel as opposed to faster. The algorithm is implemented in OpenCL and runs on the GPU, but the focus of this work will be on the algorithm, not GPU performance.

4.4 Representation

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

$$\psi(x) = \min_l |\varphi_l(x)|. \quad (4.1)$$

Figure 26 depicts both the unsigned distance field and label image representing 5 objects that overlap to create a total of $L = 8$ different object regions.

![Figure 26: (a) Distance field $\psi(x)$ and (b) label image $\chi(x)$.](image)
An approximation to $\varphi_l(x)$ is computable from $\chi(x)$ and $\psi(x)$ at the boundary of region $l$:

$$\Lambda_l = \{x| \exists y \in \mathcal{N}(x) \text{ s.t. } \sigma_l(x) \neq \sigma_l(y)\}$$  \hspace{1cm} (4.2)

where $\mathcal{N}(x)$ is the set of 2D neighbors of pixel $x$ and the sign $\sigma_l(x)$ is indicated by,

$$\sigma_l(x) = \begin{cases} -1 & \chi(x) = l \\ 1 & \text{otherwise} \end{cases}$$  \hspace{1cm} (4.3)

The partially reconstructed level set $\tilde{\varphi}_l : \Lambda_l \mapsto \mathcal{R}$ is given by,

$$\tilde{\varphi}_l(x) = \sigma_l(x) \psi(x).$$  \hspace{1cm} (4.4)

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 relative to the original level sets $\varphi_l(x)$ because we use the minimum level set value at $x$ among all objects in eq. 1. $\tilde{\varphi}_l$ provides enough information to extract an iso-surface with marching cubes [162] or recover the entire signed distance field with fast-marching [157]. 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 $\Lambda_l$ for each level set $\varphi_l$. The sparse-field LSM [158] has exactly this property.

### 4.5 Level Set Evolution

The boundaries of $L$ objects represented by $L$ level sets $\tilde{\varphi} = \{\varphi_1(x), \cdots, \varphi_L(x)\}$ are evolved by solving the following differential equation:

$$\frac{\partial \varphi_l(x,t)}{\partial t} = f_l(x,t) \delta(\varphi_l),$$  \hspace{1cm} (4.5)
where $\vec{f}(x, t) \in \mathbb{R}^L$ is the speed function and $\delta(\cdot)$ is an $L \times L$ diagonal matrix whose diagonal entries are compactly supported approximations to the dirac delta function $\delta(\cdot)$. The $i^{th}$ diagonal entry $\delta_{ii}(\vec{\varphi}) = \delta(\varphi_i)$. Subsequent examples use speed functions of the following form:

$$f_i(x) = \lambda_p \rho(x) \|\nabla \varphi_i(x)\| + \lambda_u \vec{u}(x) \cdot \nabla \varphi_i(x) + \lambda_k \kappa_i(x) \|\nabla \varphi_i(x)\|,$$  

(4.6)

where $\rho(x) \in \mathbb{R}$ is a pressure force that drives the object's boundary towards a particular image intensity, $\vec{u}(x) \in \mathbb{R}^d$ is an external velocity field that drives the boundary towards edges in the image, and $\kappa_i(x) \in \mathbb{R}$ is the mean curvature for object $l$, defined as $\kappa_i = \nabla \cdot \frac{\nabla \varphi_i}{\|\nabla \varphi_i\|}$. A common choice for pressure is $\rho(x) = l(x) - \pi$ where $\pi$ is a target intensity in image $l(x)$. Another common choice is to use the gradient of edge strength for the external velocity field $\vec{u}(x) = \nabla g(x)$ where edge strength is defined as $g = \frac{1}{1 + |\nabla|}$. Relative contributions of each force are controlled by weights $\lambda_p$, $\lambda_u$, and $\lambda_k$. Forces are computed with first-order upwind finite differences [66] on either a $3 \times 3$ or $3 \times 3 \times 3$ stencil in 2D or 3D respectively. First order, explicit integration of eq. 5 yields the following iterative scheme:

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

(4.7)

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 $\vec{\varphi}(x)$, the result must be stored in the label and unsigned distance images $\chi(x)$ and $\psi(x)$ respectively. To do this, we use the projection method proposed by Losasso et al. [69]. New level set values are sorted to find the smallest two $\varphi_a(x)$ and $\varphi_b(x)$ s.t. $\varphi_a(x) < \varphi_b(x)$ among those labels $l$ for which $x \in \Lambda_l$. The label and unsigned distance images are updated via:

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

(4.8)

and

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

(4.9)
This projection technique has several useful properties. First, it reduces the complexity of topological relationships between \( L \) objects in any finite dimension to just two objects in one dimension. Second, it ensures that objects close to 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 (Figure 27).

![Figure 27](image)

**Figure 27:** A series of active contour time steps showing how projection is used to resolve air-gaps and overlaps between objects. (a) Initial image, (b) advection, (c) projection, (d) advection, (e) projection.

The initial segmentation may have overlaps even when 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 we set a 1 in the \( k^{th} \) bit position of label \( l \). A label image stored as 32 bit integers can distinguish between 32 distinct objects and \( 2^{32} \) different combinations of objects. To scale efficiently to hundreds or thousands of objects, another option is to dedicate a single label for overlapping regions. It is necessary with either representation scheme to define forces that contract overlapping object regions so that the final segmentation contains only distinct objects.

To evolve \( \Phi_l(x) \), we evaluate eq. 7 on the subset \( \Gamma_l \):

\[
\Gamma_l = \{ x \in \Omega | -0.5 \leq \Phi_l(x) \leq 0.5 \} \subseteq \Lambda_l
\]

by multiplying \( \delta(\cdot) \) by the indicator function \( I_l(x) \), which is one if \( x \in \Gamma_l \) and zero otherwise. The CFL condition \[163\] is enforced by choosing \( \Delta t \) s.t. \( \Delta t \leq 0.5/f_{\text{max}} \) where \( f_{\text{max}} = \max_{x} | f_l(x) \delta(\psi(x,t)) | \).

To compute \( \bar{f}(x,t) \) with finite differences, \( \varphi_l(x) \) must be known in the neighborhood around \( x \) (Figure 28). If \( x \in \Lambda_l \) (point \( x_1 \) in Figure 28) and \( \chi(x) = l \), it is true that \( \forall y \in \mathcal{N}(x), \varphi_l(y) = l \).
because either $\chi(y) \neq \chi(x)$ (point $y_4$ in Figure 28) which implies $y \not\in \Lambda_i$, or $\chi(y) = \chi(x)$ which implies $y$ is inside object $l$ (points $y_1$ and $y_2$ in Figure 28), so $\psi(y)$ must be a measurement to object $l$. If $x \in \Lambda_i$ (point $x_2$ in Figure 28), $\chi(x) = m$ and $\chi(y) = n$ (point $y_3$ in Figure 28) s.t. $n, m,$ and $l$ are all different, $\psi(y)$ could be a measurement to object $n$ instead of $l$. The problem can be resolved by explicitly computing the signed distance for $\varphi_i(y)$ using the sparse-field approximation: $\varphi_i(y) = (\psi(x) + 1)\sigma_i(y)$ if $y \not\in \Lambda_i$. To address the problem in general for larger neighborhoods, we recommend the Multi-compartment LSM [152, 154]. A solution is not implemented in this work because the problem is unnoticeable in practical examples.

Figure 28: Neighborhood diagram around point $x$.

The level set evolution process is described by Algorithm 1. The unsigned distance field $\psi$ is rebuilt (Rebuild) within a distance of $G = 3$ pixels via the fast approximation described in [158] (see Algorithm 2); in which, the distance field is approximated by layers that fall in the range $[(-G + 0.5), (G - 0.5)]$. The distance at a voxel is computed based on distance measurements at neighboring voxel locations that are in a lower layer using the $L_1$ norm. Algorithm 1 is straightforward to parallelize because each for-loop over $\Omega$ is dependent on only the 2d neighbors of each pixel. The only step that is non-trivial to parallelize is computation of $f_{\max}$, which can be done with a parallel reduction [164] in $O(M^d \log_2(M)/P)$ time. This computation can be avoided by crafting $\tilde{f}$ s.t. $f_i(x, t) \in [-1, 1]$ or clamping $\tau_i(x) = f_i(x, t)\delta \left(\varphi_i(x, t)\right)$ to the range $[-1, 1]$ in Algorithm 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 \textbf{Evolve} is then \(O(M^d/P)\) for \(M^d\) pixels and \(P\) processing units.

\textbf{Algorithm 1. Evolve}

\begin{verbatim}
foreach \(x \in \Omega\) do //for each voxel in the image domain
  //Determine if \(x\) lies on a boundary \(\Gamma_t\) (eq. 10).
  if \(\exists l\ s.t. \ x \in \Gamma_l\) then
    // Compute speeds for all pixels in the active set for every pair of adjacent objects.
    foreach \(k\ s.t. \ y_k \in (\mathcal{N}(x) \cup \{x\})\) do
      // Get current object label \(l\).
      \(l = \chi(y_k, t)\)
      // Compute speed \(f_l(x, t)\) using eq. 6.
      \(\tau_k(x) = f_l(x, t) \delta(\varphi_l(x, t))\)
      \(l_k(x) = l\)
      // Find the maximum speed among all voxels and all objects that can compete for a voxel.
      \(f_{max} = \max_{k,x} |\tau_k(x)|\)
      // Determine appropriate time step that enforces CFL condition.
  if \(f_{max} > 1\) then \(\Delta t = 0.5/f_{max}\) else \(\Delta t = 0.5\)
  foreach \(x \in \Omega\) do
    if \(\exists l\ s.t. \ x \in \Gamma_l\) then
      //temporary set list
      \(Z = \emptyset\)
      // Compute level set updates for objects that compete for pixel \(x\).
      for \(k = 1:2d + 1\) do
        \(n = l_k(x)\)
        \(z = \psi_n(x, t) + \Delta t \varphi_n(x)\)
        \(Z = Z \cup \{(z, n)\}\)
      // Resolve air-gaps and overlaps with projection technique (eqs. 8 and 9).
      \textbf{Sort} \(Z\) by \(z\) to find \(\varphi_a(x)\) and \(\varphi_b(x)\)
      // Store the new level set value.
      \textbf{Compute} \(\chi(x, t + \Delta t)\) and \(\psi(x, t + \Delta t)\)
  \end{verbatim}

\textbf{Algorithm 2. Rebuild}

\begin{verbatim}
for \(g = 1:G\) do
  foreach \(x \in \Omega\) do
    // if \(x\) is not in the previous active set and the new distance measurement has not been computed yet.
    if \(\psi(x, t) > 0.5\) and \(\psi(x, t + \Delta t) > (g - 0.5)\) then
      // Estimate the minimum distance to the closest object.
      \(\psi(x, t + \Delta t) = \min_{y \in \mathcal{N}(x)} [\sigma_{\chi(x)}(y, t + \Delta t) \psi(y, t + \Delta t) - 1]\)
  \end{verbatim}
4.6 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 $512 \times 512$ “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 $L = 8$ labeled object regions. The 5 objects were evolved with pressure and curvature forces via the speed function

$$f_i(x) = \lambda_p (I(x) - \bar{I}) \| \nabla \varphi_i(x) \| + \lambda_k \kappa_i(x) \| \nabla \varphi_i(x) \|,$$  \hspace{1cm} (4.11)

where $\bar{I} = 0.5$, $\lambda_p = 0.5$, and $\lambda_k = 1$.

Overlapping object regions were assigned a constant inward pressure force ($f_i(x) = 1$) to cause their contraction. The final segmentation contains only the 5 objects with no gaps or overlaps (Figure 29b).

Figure 29: (a) Initial segmentation consisting of 5 objects. The star shaped object overlaps three other objects, which creates a total of 8 object regions. Each region is assigned a unique label in the image $\chi(x)$. (b) final image after active contour 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. Figure 30a 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 Figure 30b. Computation time is almost constant as a function of the number of objects.

Figure 30: Segmentation of “X” image as a function of (a) image size and (b) number of initial objects.

MOGAC was applied to cell tracking in a 1024 × 1024 microscopy image acquired from the Cell Centered Database (CCDB) [165]. The image was first automatically segmented with MIPAV [41] through gray-level morphology and thresholding techniques (Figure 31a). The segmentation was then refined with MOGAC to better localize boundaries on the 138 detected objects (Figure 31b). The segmentation was driven by pressure forces and external velocity field $\vec{v}(\mathbf{x})$ produced by Gradient Vector Flow (GVF) [166]. The following speed function was used

$$f_i(\mathbf{x}) = \lambda_\rho (I_B(\mathbf{x}) - \bar{I}_B)||\nabla \varphi_i(\mathbf{x})|| + \lambda_v \vec{v}(\mathbf{x}) \cdot \nabla \varphi_i(\mathbf{x}) + \lambda_k \kappa_i(\mathbf{x}) ||\nabla \varphi_i(\mathbf{x})||, \quad (4.12)$$

where $I_B$ is the blue intensity channel, $\lambda_\rho = 1$, $\bar{I}_B = 0.05$, $\lambda_v = 0.1$, and $\lambda_k = 1$. The GPU implementation of MOGAC ran for 250 iterations at a speed of 12 ms per iteration (3.03 sec. total).
Figure 31: (a) Initial segmentation of microscopy image and (b) MOGAC segmentation refinement of 138 objects. Cells appear blue in this imaging modality.

4.7 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 [102] 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 (eq. 11 with the same parameter choices) to produce a segmentation consisting of deformed versions of the torus and sphere without gaps or overlaps (Figure 32). The segmentation required 700 iterations, and the GPU implementation required, on average, 196 ms per iteration (137 sec. total).
Figure 32: (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 Figure 33a follow the same linear trends observed in the 2D case.

Figure 33: (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.

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 Figure 34 required 500 iterations, and the GPU implementation required 200 ms per iteration (100 sec. total).
Figure 34: 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 manual segmentation of the epicardium in a $128 \times 256 \times 128$ MR image that contains small gaps and overlaps between structures (Figure 35a and Figure 36a). 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 (Figure 35b and Figure 36b). The speed function in eq. 11 was used with $I = 0.8$, $\lambda_p = 1$, and $\lambda_K = 0.5$. The MOGAC clean-up required 50 iterations (2.5 sec. total).
Figure 35: 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. We thank Dr. Terry Peters for providing data.

Figure 36: Epicardium showing right ventricle and atrium (green), myocardium (blue), left ventricle (pink), and left atrium (red). (a) Initial segmentation and (b) MOGAC segmentation. We thank Dr. Terry Peters for providing data.

In the second experiment, a whole brain was segmented into 10 structures with TOADS [167] on a 256 × 256 × 256 MR image from the OASIS database [168]. The hard classification was smoothed with MOGAC for 10 iterations (2.3 sec. total) to produce the segmentation shown in Figure 37.
4.8 Discussion

A Multi-object LSM has been presented that can segment \( L \) objects with the same small memory footprint. The algorithm was applied to image segmentation in 2D and 3D. The 2D version can run at 400 Hz for any number of objects at an image resolution of \( 512 \times 512 \), making it useful for real-time computer vision applications. The 3D version can segment images faster than the Multi-compartment LSM [154], 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 7 summarizes the theoretical performance for each Multi-object LSM. MOGAC is more efficient and requires less memory than existing methods. Because the MOGAC algorithm is structurally similar to single object LSM, GPU accelerated LSM techniques described in other works are applicable [159].

Algorithm 1 spends a lot of computation time checking if the voxel belongs to the boundary of an object (\( \exists n \text{ s.t. } x \in \Gamma_n \)). To avoid traversing the entire volume, a more work-efficient approach is to index \( \Gamma_n \). A parallel algorithm for indexing \( \Gamma_n \) has already been described [159], and a
variant of that algorithm is implemented in the open-source release of MOGAC. By indexing \( \Gamma_n \), the computational complexity drops to \( O(M^{d-1}/P) \).

**Table 7: Algorithm complexity for Multi-object LSMs based on \( M^d \) pixels, \( L \) objects, and \( P \) processing units in dimension \( d = \{2, 3\} \).**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N ) level set methods ([69, 151, 153])</td>
<td>( O(LM^d \log M) )</td>
<td>( O(NM^d) )</td>
</tr>
<tr>
<td>Multi-phase ([68])</td>
<td>( O(M^d \log M \log L) ) ( O(M^d \log L) )</td>
<td></td>
</tr>
<tr>
<td>Multi-compartment ([152, 154])</td>
<td>( O(M^d \log(M/L)) )</td>
<td>( O(M^d) )</td>
</tr>
<tr>
<td>MOGAC</td>
<td>( O(M^d/P) )</td>
<td>( O(M^d) )</td>
</tr>
<tr>
<td>Work-efficient MOGAC</td>
<td>( O(M^{d-1}/P) )</td>
<td>( O(M^d) )</td>
</tr>
</tbody>
</table>

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 Figure 34). To remove these artifacts, we recommend either smoothing the final iso-surfaces in a post-processing step \([169]\) or rendering objects with volumetric techniques that reduce the appearance of artifacts \([170]\).

**Contributions:**

- The "label mask + distance field" data structure merges level sets and label masks into one representation.
- The new data structure can represent multiple objects with just two images, which is the smallest memory footprint that has been proposed for multi-object level sets.
- A multi-object level set method has been presented that is theoretically faster than all existing methods.
- The MOGAC algorithm is parallelized and implement in OpenCL to leverage GPU technology.
Chapter 5

5 Spring Level Sets (SpringLS)

5.1 Motivation

In the previous chapter, we presented a data structure that merges level sets and label masks into one representation. In this chapter, we present the Spring Level Set data structure that merges triangle meshes and level sets into one representation. The key idea is to a constellation of disconnected triangular surface elements to define a level set. These surface elements are coupled with a level set function that is evolved to track the moving constellation. The level set provides a watertight representation of the object and aids in re-sampling the constellation. MOGAC and Spring Level Sets will be combined in the subsequent chapter to create a multi-object version of Spring Level Sets that has properties of meshes, level sets, and label masks. To motivate the need for Spring Level Sets, we start by reviewing meshes, level sets, hybrid representations, and the advantages / drawbacks of each.

5.2 Background

5.2.1 Meshes

Meshes were the earliest representation for deformable models [29]. In this framework, the model is deformed by perturbing mesh vertices. The model’s boundary is explicitly tracked by remembering the trajectory of each vertex. As a section of the mesh expands or contracts, sharp creases, edges, self-intersections, or triangle flips can develop. Sharp edges and other mesh artifacts violate a common material property that most objects represented by deformable models are smooth and plastic. To reduce artifacts, the mesh must be regularized and re-sampled (re-meshed) periodically. Re-meshing is challenging (see Wojtan et al. [171] for an in-depth discussion) because triangles must be connected to form a watertight model, and re-meshing interferes with vertex tracking in non-obvious ways [46, 48, 50, 172, 173]. These
challenges have motivated the creation of new deformable model representations (e.g. marker level sets and surfels) that do not require re-meshing, handle topology changes easily, cannot self-intersect, and track moving points instead of mesh vertices.

5.2.2 Level Sets

The level set method \([66, 67]\) represents a deformable model as a 3D image where the image intensity at each voxel is the distance to the surface of the object. Distance measurements are signed: negative values are inside and positive values are outside the object. A triangle mesh can be extracted by computing the iso-surface corresponding to the zero level set of the image. The level set representation has several advantages over deformable meshes: 1) no need for self-intersection removal; 2) topology change is easy; 3) no need to re-mesh. These properties have made level sets the popular choice for image segmentation and fluid-like non-rigid deformation.

Some recent re-meshing schemes \([171, 172, 174]\) construct a level set to aid in re-meshing. Note that since the watertight property is provided by the level set in these methods, it may not be necessary to have a watertight mesh. This observation is one motivation for Spring Level Sets.

Level sets are difficult to use for registration and tracking tasks because there is no innate ability to track vertices as in the mesh deformation framework. The surface only exists when an iso-surface is extracted from the level set. Furthermore, the level set is stored as an image that is re-sampled at each time step. Re-sampling an image acts as a low-pass filter that results in feature loss as a function of the number of time steps, even if the motion is rigid (i.e. global registration) or divergence free.

5.2.3 Hybrid Representations

Attempts have been made to unify deformable model representations with varying success \([175-182]\). Of these, the Marker Level Set (MarkerLS) method \([179]\) is closest to this work. The MarkerLS method maintains a set of particles located on the level set’s zero iso-level. Since particles lie exactly on the zero iso-level, they can be used for tracking the model’s boundary. After each level set and particle advection step, the level set is corrected so that particles
continue to lie on the level set’s zero iso-level. Particles are added to cover the zero iso-level and deleted to prevent over-sampling. The MarkerLS method associates a correspondence point with each particle and interpolates the correspondence for new particles based on their neighbors.

The difference between Spring Level Sets (SpringLS) and MarkerLS is that SpringLS define the model’s level set, whereas MarkerLS use particles to correct errors in the level set. MarkerLS requires the deformation method to have an equivalent level set and parametric interpretation in order to deform both representations. Movement of the auxiliary level set with SpringLS is passive and independent of the deformation method. SpringLS can be applied more broadly to deformation methods for which there is only a parametric interpretation (e.g. Point Distribution Models (PDM) [37]), enabling true interoperability between methods designed for level sets and meshes.

5.3 Overview

The Spring Level Set (SpringLS) representation merges meshes and level sets into a single geometric representation that preserves the strengths of both. The key idea is to couple a constellation of disconnected triangular surface elements (springls) with a level set that tracks the moving constellation. Attractive forces between springls hold the constellation together, and methods are provided for re-sampling the constellation as the model deforms. Spring Level Sets (SpringLS) can be interpreted as a mesh or level set. The representation is intended to be used for methods that employ a mixture of mesh and level set techniques, but it is applicable to almost all deformable model methods.

To demonstrate the abilities and geometric properties of SpringLS, we will apply them to tasks that require the deformable model to behave as three different types of materials: rigid, elastic, and plastic. Our focus will be on the deformable model’s performance on imaging tasks. Three comprehensive applications will be described that require a mixture of deformable model methods as in the CIP (Figure 3). These include atlas-based pelvis segmentation, motion tracking in 4D CT, and morphing of articulated characters. We begin by describing the SpringLS representation, deformation process, and its implementation.
5.4 Representation

A springl is a triangular surface element consisting of a particle and three springs connecting the particle to each of the triangle’s vertices (see Figure 38 for a 2D visualization and Figure 39 for a 3D visualization).

A springl \((S_n)\) is represented as five points describing the particle location \(p_n\), correspondence point location \(a_n\), and triangle vertices \(q_{n,m}\) (Figure 39a) A correspondence point is a mapping between \(p_n\) on the current surface and a point \(a_n\) on the original surface. The clamped unsigned distance function \(d_n(X)\) is computed and also associated with each springl. The distance field is
clamped at $d_{max} = 0.5$ voxels. The combination of these distance functions form the unsigned level set $\omega(X)$ (Figure 38b). The support of a springl is represented by a capsule (green region in Figure 39), whose boundary is the iso-surface corresponding to $d_{max}$ (black outline in Figure 39). The union of all capsules creates a thick boundary representation. Springs, vertices, and particles are coplanar, and the angles between springs are fixed. An auxiliary signed level set $\varphi(X)$ is maintained and evolved with the particles (Figure 38b). The level set augments the particle representation in several ways: 1) the signed distance function indicates regions that are inside or outside the model; 2) the signed level set indicates when new springls need to be added or removed; 3) the iso-surface extracted from the signed level set is a watertight triangle mesh representation of the model. The deformable model is stored as three data structures: a collection of triangles representing surface elements ($q_{n,m}$), a point cloud of particles ($p_n$), and a 3D image representing the signed level set ($\varphi(X)$).

5.5 Deformation Algorithm

5.5.1 Advection

The model is deformed by incrementally advecting springls with Lagrangian methods. The deformation may be driven by pressure $\rho(\cdot)$ in the normal direction $\vec{n}_n$ of a springl, an external velocity field $\vec{v}(\cdot)$, and/or a local affine transformation $A(\cdot)$ that is a function of the particle position and its corresponding point $a_n$ on the initial object:

$$\frac{\partial p_n}{\partial t}(t) = \lambda_p \vec{n}_n \rho(p_n(t)) + \lambda_v \vec{v}(p_n(t)) + \lambda_A A(p_n(t), a_n)p_n(t), \quad (5.1)$$

And

$$\frac{\partial q_{n,m}}{\partial t}(t) = \lambda_p \vec{n}_n \rho(p_n(t)) + \lambda_v \vec{v}(p_n(t)) + \lambda_A A(p_n(t), a_n)q_{n,m}(t). \quad (5.2)$$

See Table 8 for definitions of terms. Note that vertices are advected based on velocities evaluated at the particle location. The advection equations (eq. 1 and 2) are sufficient for modeling a broad spectrum of materials. Fluid deformation methods use pressure ($\rho$) and
external velocity field ($\vec{v}$), whereas rigid and elastic deformation methods use local affine transformations ($A$).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_n$</td>
<td>Particle location.</td>
</tr>
<tr>
<td>$\alpha_n$</td>
<td>Corresponding particle location on initial surface.</td>
</tr>
<tr>
<td>$q_{n,m}$</td>
<td>Triangle vertex $m = {0,1,2}$.</td>
</tr>
<tr>
<td>$\vec{n}_n$</td>
<td>Triangle normal.</td>
</tr>
<tr>
<td>$S_n$</td>
<td>$4 \times 4$ matrix describing a springl.</td>
</tr>
<tr>
<td>$q_{n,m,k}$</td>
<td>$k^{th}$ closest point on nearby triangle edge from $q_{n,m}$.</td>
</tr>
<tr>
<td>$d_{\text{max}}=0.5$</td>
<td>Springl capsule radius.</td>
</tr>
<tr>
<td>$r = 0.05$</td>
<td>Particle and triangle vertex radius.</td>
</tr>
<tr>
<td>$R = d_{\text{max}} + 2r$</td>
<td>Nearest-neighbor range.</td>
</tr>
<tr>
<td>$\omega(X)$</td>
<td>Unsigned level set.</td>
</tr>
<tr>
<td>$\varphi(X)$</td>
<td>Signed level set.</td>
</tr>
<tr>
<td>$\varphi_{\text{ref}}(X)$</td>
<td>Initial signed level set.</td>
</tr>
<tr>
<td>$\alpha(x)$</td>
<td>Weight function.</td>
</tr>
<tr>
<td>$\beta(x)$</td>
<td>Threshold function.</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Threshold function smoothness.</td>
</tr>
<tr>
<td>$\lambda_p$</td>
<td>Pressure force weight.</td>
</tr>
<tr>
<td>$\lambda_a$</td>
<td>Advection force weight.</td>
</tr>
<tr>
<td>$\lambda_A$</td>
<td>Affine transformation weight.</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Level set regularization weight.</td>
</tr>
<tr>
<td>$\kappa_{\text{int}}$ and $\kappa_{\text{ext}}$</td>
<td>Spring constants for internal and external forces.</td>
</tr>
<tr>
<td>$\xi = q = 1$</td>
<td>Step sizes for relaxation.</td>
</tr>
<tr>
<td>$M(\vec{b}_n)$</td>
<td>Rotation matrix representing the moment $\vec{b}_n$.</td>
</tr>
<tr>
<td>$VE(S_n)$</td>
<td>Local potential energy of springl $n$.</td>
</tr>
<tr>
<td>$\mathbb{I}(x \leq y)$</td>
<td>Function that is 1 for $x \leq y$ and 0 otherwise.</td>
</tr>
</tbody>
</table>

5.5.2 Relaxation

**Spring-Mass System**

After each advection step, particles are fixed and the shape/orientation of springls are adjusted. To understand the relaxation process, we first introduce the system of forces as a spring-mass system. Springls are only permitted to rotate around their particle and adjust their spring lengths, so we re-express vertices in a local coordinate system as vertices $\vec{q}_{n,m} = q_{n,m} - p_n$, spring lengths $l_{n,m} = \|\vec{q}_{n,m}\|$, and tangent vectors $s_{n,m} = M(\vec{b}_n)q_{n,m}/l_{n,m}$. A springl configuration can be transformed back into world coordinates through
\[ q_{n,m} = l_{n,m}M(b_n)\vec{s}_{n,m} + p_n. \]  

\[ M(b_n) \] is a rotation matrix corresponding to the springl's pose, represented by an axis-angle vector \( b_n \in \mathbb{R}^3 \). Thus, \( l_{n,m} \) and \( b_n \) express the 6 degrees of freedom each springl has during relaxation. Springls have an internal spring force \( \vec{f}^\text{int}_{n,m} \) that attracts vertices along their tangent vectors \( \vec{s}_{n,m} \) towards the particle \( p_n \)

\[ \vec{f}^\text{int}_{n,m} = \kappa_{\text{int}}(l_{n,m} - 2r)M(b_n)\vec{s}_{n,m}, \]  

where \( \kappa_{\text{int}} \) is the internal spring constant. To prevent triangle flips and unstable normals, the resting spring length is chosen to be \( 2r \). An external spring force \( \vec{f}^\text{ext}_{n,m} \) attracts vertices to the edges of nearby triangles

\[ \vec{f}^\text{ext}_{n,m} = \kappa_{\text{ext}}\sum_k(q_{n,m} - q_{n,m,k}), \]  

where \( \kappa_{\text{ext}} \) is the external spring constant and \( q_{n,m,k} \) is the closest point to \( q_{n,m} \) on the perimeter of the \( k \)th neighboring springl within a radius \( R \) from particle \( p_n \) (Figure 40). To detect whether a springl is within a radius of \( R \) from a vertex, only the particle location for the springl is checked. This was done for computational efficiency.

\[ \text{Figure 40: Diagram of 3D springl. Gray squiggles indicate springs.} \]

Relaxation seeks to find a set of spring lengths \( l_{n,m} \) and pose \( b_n \) that place the system in equilibrium. Equilibrium occurs when the gradient of potential energy
\[ VE = \sum_n \sum_m \sum_k \frac{1}{2} \kappa_{\text{ext}} \| q_{n,m} - q_{n,m,k} \|^2 + \sum_n \sum_m \frac{1}{2} \kappa_{\text{int}} (l_{n,m} - 2r)^2 \]  \hspace{1cm} (5.6)

is zero. Global optimization of \( VE \) is impractical because external constraints do not have a simple analytic formulation. To make optimization of \( VE \) tractable, we take a greedy approach by optimizing each spring individually while assuming that the \( q_{n,m,k} \)s are fixed:

\[ VE(S_n) = \sum_m \sum_k \frac{1}{2} \kappa_{\text{ext}} \| q_{n,m} - q_{n,m,k} \|^2 + \sum_m \frac{1}{2} \kappa_{\text{int}} (l_{n,m} - 2r)^2 \]  \hspace{1cm} (5.7)

Solving for the partial derivative of \( VE(S_n) \) w.r.t. \( l_{n,m} \) and \( \bar{b}_n \), we obtain:

\[ \frac{\partial VE}{\partial l_{n,m}} = (\mathbf{M}(\bar{b}_n) \bar{s}_{n,m}) \cdot (f_{\text{int}m} + f_{\text{ext}m}) \]  \hspace{1cm} (5.8)

\[ \frac{\partial VE}{\partial \bar{b}_n} = \sum_m l_{n,m} \bar{s}_{n,m} \times \bar{f}_{n,m} \]  \hspace{1cm} (5.9)

(see Appendix A for derivation). The gradient is used to minimize \( VE(S_n) \) via the following iterative scheme:

\[ l_{n,m}(z + 1) = l_{n,m}(z) - \xi \frac{\partial VE}{\partial l_{n,m}}(z) \]  \hspace{1cm} (5.10)

and

\[ \tilde{b}_n(z + 1) = \tilde{b}_n(z) - \rho \frac{\partial VE}{\partial \tilde{b}_n}(z) \]  \hspace{1cm} (5.11)

where \( \xi \) and \( \rho \) are appropriate step sizes. The step sizes multiply the spring constants \( \kappa_{\text{ext}} \) and \( \kappa_{\text{int}} \) in the expression for \( \frac{\partial VE}{\partial l_{n,m}} \) and \( \frac{\partial VE}{\partial \tilde{b}_n} \), so we decided to fix the step sizes at \( \xi = \rho = 0.1 \) and only tune the free parameters \( \kappa_{\text{ext}} \) and \( \kappa_{\text{int}} \). In an attempt to minimize the global potential
energy, the $q_{n,m,k}$s are recomputed after each iteration of eqs. 10 and 11 under the constraint that points can only slide along the triangle edge with which they are currently associated. This was done to avoid a scenario where a spring continuously rotates if the spring is allowed to switch to a different edge on the same triangle.

Non-Physical System

There are other suitable definitions of attraction forces, for which we generalize $f_{n,m}^{\text{ext}}$ to use an arbitrary weight $\alpha(x)$ and threshold $\beta(x)$ function. The weighted external force is given by

$$\vec{c}_{n,m} = \sum_k \alpha(||d_{n,m,k}||)d_{n,m,k},$$

(5.12)

where $d_{n,m,k} = q_{n,m} - q_{n,m,k}$ is the displacement vector between the vertex and the closest edge on a triangle within a radius of $R$ from $q_{n,m}$. The thresholded external force is given by

$$\vec{g}_{n,m} = \beta(||\vec{c}_{n,m}||)\vec{c}_{n,m}.$$

(5.13)

The local potential energy equation has been modified to reflect this divergence from a physical system:

$$VE^*(S_n) = \sum_m \frac{1}{2}||\vec{g}_{n,m}||^2 + \sum_m \frac{1}{2}k_{\text{int}}(l_{n,m} - 2r)^2$$

(5.14)

The gradient of $VE^*$ is the following (see Appendix A for derivation):

$$\frac{\partial VE^*}{\partial l_{n,m}} = (M(\vec{b}_n)\vec{z}_{n,m}) \cdot (f_{n,m}^{\text{int}} + I_{n,m}^T \vec{g}_{n,m}),$$

(5.15)

and
\[
\frac{\partial V E^*}{\partial b_n} = \sum_m l_{n,m} s_{n,m} \times \left( J_{n,m}^T \tilde{g}_{n,m} \right),
\]

(5.16)

where \( J_{n,m} \) is the 3 \times 3 Jacobian of \( \tilde{g}_{n,m} \) w.r.t. \( q_{n,m} \) (see Appendix for derivation). Depending on choices for \( \alpha(x) \) and \( \beta(x) \), \( J_{n,m} \) could be very difficult to compute. Assuming \( \alpha(x) \) and \( \beta(x) \) were chosen such that \( J_{n,m} \) is close to an identity matrix, we use the following iterative scheme that is approximately gradient descent:

\[
l_{n,m}(z + 1) = l_{n,m}(z) - \xi \left( M \left( \tilde{b}_n(z) \right) s_{n,m} \right) \cdot \left( \tilde{f}_{n,m}^{\text{int}}(z) + \tilde{g}_{n,m}(z) \right),
\]

(5.17)

and

\[
\tilde{b}_n(z + 1) = \tilde{b}_n(z) - \varrho \tilde{y}_n(z),
\]

(5.18)

Where,

\[
\tilde{y}_n(z) = \sum_m l_{n,m}(z) s_{n,m} \times \tilde{g}_{n,m}(z).
\]

(5.19)

Selection of \( \alpha(x) \) and \( \beta(x) \) will be discussed in section 5.7.1 on parameter choices.

The purpose of relaxation is to smooth and regularize the constellation while ensuring that the union of all spring capsules covers the zero iso-level of the signed level set. To this end, we decided not to use physical torque. Figure 41 illustrates that if we use physical torque \( \tau_{n,m} = l_{n,m} \tilde{s}_{n,m} \times \tilde{g}_{n,m} \) as in a spring-mass system, the equilibrium configuration has increased gap size and disparity in normals and spring lengths. Normalizing torque \( \tau_{n,m}^* = \tilde{s}_{n,m} \times \tilde{g}_{n,m} \) reduces gap size and improves the uniformity of spring lengths. Consequently, \( \tilde{y}_n \) has been changed to reflect this modification:
\[ \vec{y}_n^* (z) = \sum_m \vec{\tilde{z}}_{n,m} \times \vec{g}_{n,m} (z). \] (5.20)

Figure 41: (a,b) Initial springl configuration and equilibrium configuration with (c) physical torque \( \tau_{n,m} \) and (d) normalized torque \( \hat{\tau}_{n,m} \). Arrows indicate perpendicular component of \( \vec{g}_{n,m} \). The springl does not rotate in (d) because \( \vec{y}_n^* (0) = 0 \) and the perpendicular component of \( \vec{g}_{n,m} \) is constant.

5.5.3 Filling Gaps

We want to ensure that the union of all springl capsules covers the zero iso-level of the signed level set. If springls are unable to cover the zero iso-level through relaxation, gaps must be filled by adding more springls in a subsequent step. This is done so that the surface remains densely sampled with springls, which is important for point tracking. When a zero-crossing of the level set is exposed, a springl is added to cover the zero-crossing (Figure 42). Zero-crossings are evaluated at the centroids of triangles generated from the signed level set’s iso-surface via Marching Cubes [162]. These triangles are reused to fill the hole with a springl in the same shape and position. In doing so, the maximum gap between springls cannot not exceed a sphere of radius \( d_{max} \).
Figure 42: (a) Exposed zero-crossing of iso-contour is detected and (b) zero-crossing is covered by inserting a springl.

5.5.4 Level Set Evolution

The signed level set provides a redundant, implicit representation of the model, which is parameterized by a constellation of springsl. To enforce consistency between the constellation and signed level set, the level set is evolved to track the moving constellation. This is done by constructing an unsigned level set \( \omega(X) \) (eq. 23), which is the clamped minimum distance to all springsl.

\[
\omega(X) = \min\{2d_{\text{max}}, d_1(X), \ldots, d_N(X)\}, \tag{5.21}
\]

The signed level set is then evolved to minimize the energy function in eq. 24 (See [66, 67] for a detailed description of level set evolution).

\[
E = \int \left( \frac{1}{2} \omega(X)^2 + \lambda |\nabla \omega(X)| \right) \delta(\omega(X)) dX, \tag{5.22}
\]

where the free parameter \( \lambda \) controls the model’s smoothness. The energy function is minimized with the following iterative scheme where \( \delta_\epsilon(x) \) is a regularized dirac delta function.

\[
\phi^{x+1}(X) = \phi^x(X) - \Delta t \delta_\epsilon(\phi^x(X)) \left( \omega(X) \nabla \omega(X) \cdot \frac{\nabla \phi^x(X)}{|\nabla \phi^x(X)|} + \lambda \nabla \cdot \frac{\nabla \phi^x(X)}{|\nabla \phi^x(X)|} \right). \tag{5.23}
\]

Level sets can change topology when a portion of the model thins below the resolution of the grid. This is sometimes undesirable if the object has a known, fixed topology. Instead of increasing grid resolution in thin regions [89], the level set method can be augmented to explicitly preserve topology [183]. The idea is to prevent a voxel's level set value from changing when a topology change is about to occur. However springsl will continue to move, possibly far
from the zero iso-level. The advection equations have been modified to stop springl movement when the topology constraint prevents the zero iso-level from moving:

\[
\frac{\partial p_n}{\partial t}(t) = \begin{cases} 
\frac{\partial p_n}{\partial t}(t), & \left(\varphi(p_n(t)) - \varphi(p_n(t - 1))\right) \left(\nabla \varphi(p_n(t)) \cdot \mathbf{n}_n(t)\right) \geq 0 \\
0, & \text{otherwise} 
\end{cases}
\]

\[
\frac{\partial q_{n, m}}{\partial t}(t) = \begin{cases} 
\frac{\partial q_{n, m}}{\partial t}(t), & \left(\varphi(p_n(t)) - \varphi(p_n(t - 1))\right) \left(\nabla \varphi(p_n(t)) \cdot \mathbf{n}_n(t)\right) \geq 0 \\
0, & \text{otherwise} 
\end{cases}
\]

(5.24)

(5.25)

For large elastic deformations where the CFL number exceeds 1, it is usually faster to convert the unsigned level set to a signed level set (\textbf{UnsignedToSigned}) than to evolve the signed level set with active contour methods. This problem can be formulated as watertight surface reconstruction from point clouds, for which there has been extensive research [184-190]. However, we can use a simpler algorithm based on region growing because our point cloud data has no noise.

We perform the unsigned-to-signed conversion quickly by growing the background region (outside the object) and negating the unsigned level set in the foreground region (complement of background) (see Figure 43). This is done robustly with a coarse-to-fine strategy to prevent the background region from leaking through gaps between springls. To do so, the background is first grown for 16 iterations on a coarse grid (16 × 16 × 16), starting from the boundary of the volume. The grid resolution is then doubled, and the background is grown for 2 iterations. The process is repeated until the grid resolution matches the resolution of the level set image. This technique provides a fast coarse estimate of the object’s boundary and is robust to gaps between springls. After initial unsigned-to-signed conversion, \textbf{Evolve} is used to accurately minimize eq. 24.
Figure 43: Unsigned-to-signed conversion by region growing. The unsigned distance field is depicted as a gray-level image. The background region first grown on an 8x8 grid (blue). The resolution is then increased to 16x16 (red), 32x32 (green), 64x64 (yellow) until the region becomes a good approximation to the object's boundary. The unsigned distance field is then negated in the foreground region and the object's boundary is accurately localized with level set evolution.

5.5.5 Contraction and Re-sampling

Springs are re-sampled to regularize the sampling distribution and triangle quality. Triangles are split along their longest edge if the length of that edge exceeds a threshold (1.5 voxels) (Re-sample). If a triangle’s angles fall outside a tolerable range $[20^\circ, 160^\circ]$, then the springl is removed (Contract) to reduce anisotropic sampling of the underlying manifold surface. A springl is destroyed if after evolving the signed level set a springl's particle is more than $(1 + \varepsilon)d_{\text{max}}$ from the zero iso-level of the signed level set (Contract). We choose $\varepsilon = 0.25$ in all cases. Figure 44 depicts an example of removing springls that lie too far from the zero iso-level.
Figure 44: Diagram showing the collision between two circular objects and the merging of their springls constellations. Note that in this example, using either UnsignedToSigned or level set evolution to track the constellation will produce the same result. UnsignedToSigned will reconstruct a signed level set that is the envelope around the model, and level set evolution will merge the objects as a consequence of enforcing the entropy condition [157]. After updating the level set, Contract will remove interior springls.

5.5.6 Tracking

SpringLS maintain a mapping from each particle to the centroid of a triangle on the original model. The initial mapping is an identity mapping \((a_n = p_n)\). When a springl is split, the mapping is duplicated and when a springl is added, the mapping is chosen to be either the average point mapping for neighboring springls within a radius of \(R\) or the point mapping for the closest springl within a radius of \(R\). If the distance between the average and nearest point correspondence is larger than a threshold (2 voxels), then the nearest point correspondence is used instead of the average. This makes the method robust to situations where neighboring point correspondences are far from each other on the original object, resulting in an average mapping that is far from the object's boundary. Still, this method can produce mappings that lie slightly off the original surface. To prevent correspondence points from drifting from the original surface, the correspondence point is moved along the gradient of the distance \((\varphi_{ref})\) to the original surface:

\[
a_n(t + 1) = a_n(t) - \lambda \varphi_{ref}(a_n(t)) \nabla \varphi_{ref}(a_n(t)),
\]

(5.26)

for 4 iterations \((\lambda = 0.5)\). After which, \(a_n\) can be projected onto the closest triangle on the original surface and referenced with barycentric coordinates.
5.6 Implementation

Since SpringLS requires updating both a particle system and level set, it has the computational overhead of both. Fortunately, there are solutions for parallelizing particle systems and level sets that enable them to run at interactive frame rates [159, 191, 192]. SpringLS reuses these parallel design patterns and is implemented in OpenCL, a programming language intended for multi-core architectures, such as the GPU. Modern CPUs also support OpenCL. We prefer to run OpenCL on the CPU because the CPU can access more global memory, which allows the software to work with larger problem sizes on inexpensive hardware. It also frees up the GPU to be used for real-time volumetric rendering. Table 9 summarizes core algorithms implemented in SpringLS.

Table 9: Description of core algorithms used in Spring Level Sets.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advect</td>
<td>Apply forces to move springls.</td>
</tr>
<tr>
<td>Relax</td>
<td>Regularize springls constellation.</td>
</tr>
<tr>
<td>Evolve</td>
<td>Evolve signed level set to track springls.</td>
</tr>
<tr>
<td>Rebuild</td>
<td>Rebuild the signed distance field.</td>
</tr>
<tr>
<td>Contract</td>
<td>Remove poor quality springls and those far from the signed level set’s iso-surface.</td>
</tr>
<tr>
<td>Re-sample</td>
<td>Split springls whose area exceeds a threshold.</td>
</tr>
<tr>
<td>FillGaps</td>
<td>Add more springls to fill holes in the constellation.</td>
</tr>
<tr>
<td>UnsignedToSigned</td>
<td>Generate a coarse reconstruction of the signed level set from the springls constellation.</td>
</tr>
</tbody>
</table>

For efficient data manipulation on OpenCL devices, a springl is stored as a 4x4 matrix, where each row corresponds to a float4 vector:

\[ S_n = \begin{bmatrix} p_n^x & p_n^y & p_n^z & \varphi_n \\ q_{n,1}^x & q_{n,1}^y & q_{n,1}^z & a_n^x \\ q_{n,2}^x & q_{n,2}^y & q_{n,2}^z & a_n^y \\ q_{n,3}^x & q_{n,3}^y & q_{n,3}^z & a_n^z \end{bmatrix} \]  \hspace{1cm} (5.27)

The signed level set value (\( \varphi_n \)) is a temporary variable used to indicate the destruction of a springl. A spatial look-up table is constructed using the uniform grid hashing procedure described in [193]; but instead of spherical particles, springl capsules are used. We prefer the implementation that uses atomics, which has \( O(N/P) \) time complexity for \( N \) springls and \( P \)
processing units. It is assumed that a grid cell contains at most 16 springl capsules, and a springl capsule falls into at most 32 grid cells. The relaxation and re-sampling phases reinforce these assumptions, but local failures may occur when these assumptions are violated. The implementation prevents the number of springl capsules stored in a grid cell from exceeding the maximum occupancy. The unsigned level set $\omega(X)$ is computed in a reduction phase that finds the minimum distance among all springl capsules that fall into a grid cell.

Springl neighbors are computed by first enumerating all springls in all hash bins that fall within a distance of $R$ from each vertex $q_{n,m}$, as determined by the particle location for nearby springls (i.e. $\|q_{n,m} - p_x\| \leq R$ where $p_x$ is the particle for a nearby springl). The list is then sorted by springl id to identify duplicates, and at most 8 springls within a distance $R$ are considered in computing $\bar{g}_{n,m}$. Nearest-neighbors are stored as tuples containing the springl id and edge id for the edge with the closest point $q_{n,m,k}$. During relaxation, the closest point can slide along an edge; but to avoid recomputing neighbors, it is assumed the closest point does not switch to a different edge or an edge on a different triangle.

The advection phase applies a displacement to each springl (see eq. 1 and 2). The maximum displacement of any springl particle cannot exceed $d_{max}$ because then the zero iso-level of the signed distance function could fall outside the capture range of the unsigned level set. This limitation can be circumvented in one of two ways. Either the unsigned distance field can be extended with fast-marching [194], or the unsigned level set can be converted to a signed level set ($\text{UnsignedToSigned}$). We do neither in the active contour deformation algorithm and instead choose a small time step ($\Delta t \leq d_{max}/f_{max}(t)$) where $f_{max}(t) = \max_n\|\frac{\partial p_n}{\partial t}(t)\|$. 

Algorithm 1: Advect

```plaintext
foreach springl n do  //The time step is chosen s.t. $\Delta t < d_{max}/|p_n(t+1) - p_n(t)|$
    $p_n(t+1) = p_n(t) + \Delta t\left(\lambda_p \bar{n}_n \rho(p_n(t)) + \lambda_\sigma \bar{\sigma}(p_n(t)) + \lambda_A(p_n(t), a_n) p_n(t)\right)$

foreach vertex m do
    $q_{n,m}(t+1) = q_{n,m}(t) + \Delta t\left(\lambda_p \bar{n}_n \rho(p_n(t)) + \lambda_\sigma \bar{\sigma}(p_n(t)) + \lambda_A(p_n(t), a_n) q_{n,m}(t)\right)$
```

The relaxation phase computes new springl configurations based on the current constellation and then applies the updates in a second pass to generate springl configurations for the next iteration. The process is repeated for 20 iterations (see Algorithm 2).
Algorithm 2: Relax

```plaintext
foreach springl n do
    foreach vertex m do
        foreach neighbor k of vertex m do
            // compute weight for each vertex and its closest point on a nearby springl.
            Compute $\alpha(q_{n,m}, q_{n,m,k})$ (eq. 12 and Figure 45)
            // accumulate total force for each triangle vertex.
            Accumulate $\vec{c}_{n,m}$
            // compute total external and internal force.
            Compute $\vec{g}_{n,m}$ and $\vec{s}_{n,m}$
            // accumulate normalized torque.
            Accumulate $\vec{r}_{n,m}$
            // apply apply updates to springl configuration.
        foreach vertex m do
            Compute $q^{t+1}_{n,m}$
```

Advection and relaxation are trivial to parallelize because there are a small number of data dependencies and the dependencies are read-only. Contraction and expansion are more difficult because removing or inserting elements in an array of springls can affect the contents of the entire array. The solution is to split the task into two steps. First, the number of contracted or inserted elements is counted. A parallel implementation of a prefix sum [195] is used to compute the new array indices for the springls, and springls are either copied into or created at new positions in the array.

SpringLS uses an implementation of the sparse-field level set method [196] that has been parallelized. After evolving the level set with eq. 23 (Evolve), the signed distance field is rebuilt (Rebuild) in the region near the model’s boundary, commonly referred to as the narrow-band. Indexes into the narrow-band are stored in a buffer to avoid traversal of the entire level set volume. The narrow-band buffer is updated after each pass of Rebuild. Since parallel narrow-band implementations are non-trivial, we refer the reader to [159, 197]. The time complexity of the parallel sparse-field algorithm is $O(M^2/P)$ for $M^3$ voxels and $P$ processing units.

Algorithm 3 describes the complete springls deformation algorithm for $K$ iterations and a re-sampling period of $M$ (i.e. the contraction and re-sampling phases are performed every $M^{th}$ iteration). We choose $M = 5$ in all cases. This algorithm is intended for fluid-like deformations,
but there are other acceptable variations on the SpringLS deformation algorithm. For instance, Algorithm 4 is a more elastic version of the deformation algorithm that permits large step sizes; or alternatively, the model behaves like a mesh (without triangle connectivity) if only Advect is used. If Contract, Resample, and FillGaps are disabled, a 1-to-1 mapping between springls is maintained. If Relax is disabled for some springls, then corresponding geometric features on the initial mesh are preserved.

**Algorithm 3: Active Contour Deformation**

```plaintext
for k = 1:K do //for each iteration k
    //move constellation.
    Advect
    //relax constellation to reduce gap size.
    Relax
    if k mod M == 0 then
        //re-sample constellation
        Contract
        Re-sample
        Relax
        for z = 1:4 do
            //evolve level set to track moving constellation.
            Evolve
            //rebuild signed distance field in the narrow-band around the zero iso-level.
            Rebuild
            FillGaps
    else
        for z = 1:4 do
            //evolve level set to track moving constellation.
            Evolve
            //rebuild signed distance field in the narrow-band around the zero iso-level.
            Rebuild
```

---

**Algorithm 4: Elastic Deformation**

```plaintext
for k = 1:K do //move constellation.
    Advect
    UnsignedToSigned
    for z = 1:4 do
        Evolve
        //rebuild signed distance field in the narrow-band around the zero iso-level.
        Rebuild
    //relax constellation to reduce gap size.
    Relax
    //re-sample constellation
    FillGaps
    Contract
    Re-sample
```
Algorithm 4 has no CFL time step restriction because it uses UnsignedToSigned to reconstruct the signed level set at each time step. Depending on the step size, UnsignedToSigned may be more efficient than level set evolution. The choice between algorithms depends on the problem and if there is already a step size restriction due to the nature of the problem (e.g. fluid simulations).

5.7 Performance Analysis

5.7.1 Parameter Choices

There are many parameters associated with the SpringLS representation, some intrinsic to the model and others image / task dependent. When selecting intrinsic parameters, the primary concern is to choose those that hold the constellation of springs together during deformation as much as possible. The FillGaps step is a catch-all to fix holes in the constellation that occur due to advection and relaxation. However, it is desirable to minimize the amount of gap filling because it requires the introduction of more springs and interpolation of new point correspondences. Consider a pathological case where all springs are destroyed in the Contract step. In the case where $\lambda = 0$ (no regularization in eq. 24), the Evolve step will not move the interface because the unsigned level set is $d_{max}$ everywhere. The FillGaps step will then regenerate a constellation of springs that consists of all triangles in the iso-surface. Even though the deformable model was prevented from tearing, all tracking information was lost and the model failed to deform. We would like to preserve the constellation with relaxation to avoid this scenario.

Many parameters can be chosen heuristically (Table 1), leaving the spring constant $k_{int}$ and functions $\alpha(x)$ and $\beta(x)$ to be chosen empirically. For weight and threshold functions of the form $\alpha(x) = \frac{1}{x}W(x-2r)$ and $\beta(x) = \frac{1}{x}H(\lambda_c x)$, we looked at two choices for $H(x)$: $H_a(x) = x$ and $H_b(x) = \tanh(x)$, and four choices for $W(x)$: $W_a(x) = x$, $W_b(x) = \text{atanh}(x)$, and $W_c(x)$ and $W_d(x)$ are Hermite splines satisfying:
where $R$ is chosen to be larger than $d_{max}$ (see Figure 45 for depiction of functions). $\lambda_c$ is a free parameter that will be referred to as the sharpness of $H(x)$. A repulsion force appears in $\alpha(x)$ to keep springls at a minimum distance of $2r$. This is analogous to the repulsion force used in particle systems to bound the number of particles that can occupy a grid cell without overlap, and it is equivalent to having springs with non-zero spring length. The data structure used to index springls assumes there is a bound on the number of springls that can occupy a grid cell [193]. The repulsion force reinforces this assumption.

$H_a(x)$ paired with $W_a(x)$ causes the attractive forces between springls to behave like springs. A potential drawback of these functions is that an imbalance in the number of neighboring springls will cause an imbalance in the rotational forces exerted on a springl, even if the proximity of neighboring springls is the same. This is due to the sum that appears in the expression for the external force $\tilde{c}_{n,m}$ (eq. 12). To address this concern, we propose $H_b(x)$ paired with $W_b(x)$ to dampen the cumulative force that can be exerted by neighboring springls. However, both sets of functions have infinite support even though the force is only applicable to springls within a distance of $R$. To assess whether this is a problem, we propose $W_c(x)$ and $W_d(x)$; for which, the force exerted by neighboring springls smoothly decreases to zero within its neighbor range.
Figure 45: (a) $H(x)$ and (b) $W(x)$ are considered in parameter tuning experiments. These functions are related to threshold $\beta(x)$ and weight $\alpha(x)$ functions via $\beta(x) = \frac{1}{\sigma} H(\lambda_c x)$ and $\alpha(x) = \frac{1}{\sigma} W\left(\frac{1-2r}{R}\right)$.

5.7.2 Parameter Tuning

Enright et al. [175] introduced two tasks that test extremes of deformable model behavior. The first test places a notched sphere (Zalesak’s sphere) in a velocity field that rotates the object $360^\circ$ every 628 time steps. The second test places a sphere in an incompressible flow field proposed by LeVeque [198] that stretches the object into a disk. The flow is time-reversed by modulating the velocity field with $s(t) = \cos\left(\frac{\pi t}{T}\right)$. The model achieves maximal deformation at time $t = T/2$ and returns to a sphere at $t = T$, where $T$ controls the maximum amount of deformation (Figure 47). In both tests (Figure 47 and Figure 46), particle advection was integrated in time with a 4th order Runge-Kutta scheme. The deformable model should return to its initial configuration at the end of Zalesak’s and LeVeque’s test (i.e. $t = T$).

The level set method performs poorly on both tests by failing to preserve shape and volume, which is important in simulation of solids and fluids. LeVeque’s test is difficult for mesh representations because it requires re-meshing. There have been several papers that present deformable model representations which perform better than the level set method on LeVeque’s test. Wicke et al. [173] use a tetrahedral mesh, Bargteil et al. [174] use semi-Lagrangian contouring, and several other methods [90, 199, 200] use a level set coupled with a set of particles. SpringLS parameters were selected based on LeVeque’s test because it has become a popular benchmark.
Figure 46: Zalesak's sphere advected with a rotational velocity field. The initial (0º) and final (360º) constellation are almost identical. Grid dimensions were $256 \times 256 \times 256$.

Figure 47: LeVeque's test showing sphere deformed into a disk. The model reaches its maximal deformation at $t=0.5T$ and returns to a sphere at time $t=T$ for a period of $T=3$. Grid dimensions were $256 \times 256 \times 256$. Simulation used 10K - 32K springls.

For the purposes of parameter tuning, LeVeque's test was evaluated on a small grid ($128^3$) for a maximum deformation of $T = 2.5$. We looked at 4 different metrics: deformation error, tracking error, average number of springls, and total number of added and removed springls (Fig. 11). Deformation error refers to the point-to-surface distance between particle locations $p_n$ and the original implicit model $\varphi_{ref}(X)$, or simply $|\varphi_{ref}(p_n)|$. Tracking error refers to the distance
between $a_n$ and $p_n$ for each spring, or $\|p_n - a_n\|$. For each set of parameters, all 8 combinations of weight and threshold functions were evaluated. Le Veque’s test was executed on a $128 \times 128 \times 128$ grid for a maximum deformation of $T = 2.5$. Figure 48 shows the minimum total error (deformation plus tracking error) among all pairs of threshold / weight functions. $H_b(x)$ paired with either $W_a(x)$ or $W_b(x)$ perform the best. We chose to use $H_b(x)$ and $W_a(x)$ because $W_a(x)$ is faster to evaluate than $W_b(x)$. For this choice of functions, there is a sizable subset of parameters that achieve similar performance (Figure 49b). Subsequent experiments use $\kappa_{int} = 0.3$ and $\lambda_c = 5$, and none of the intrinsic parameters are changed or re-tuned. Note that non-physical relaxation uses $\lambda_c$ instead of $\kappa_{ext}$ because $\kappa_{ext}$ implies a linear spring.

![Deformation and Tracking Performance](image)

**Figure 48:** Plot of the minimal total error (deformation plus tracking error) among all pairs of threshold / weight functions. Darker regions have lower error. Glyphs indicate the best performing set of threshold / weight functions. The shape and color of each glyph corresponds to the functions in Figure 45.
Figure 49: Plot of performance metrics for (a) $H_a(x)$ & $W_a(x)$ and (b) $H_b(x)$ & $W_b(x)$. The minimum value is marked with a "\(\phi\)". Blue regions indicate lower error or number of springls, and red regions indicate higher error or number of springls.

5.7.3 Computational Performance

Performance was evaluated on a PC with Dual Intel E5630s. Parameter choices reflect the optimal ones selected in the previous section. Le Veque’s test was repeated on different size grids for a period of $T = 2.5$. Table 10 summarizes these experiments. Deformation and tracking errors are almost constant in voxel units. Figure 50 shows an example of LeVeque’s test at $384^3$ and Figure 51 shows the computational performance as a function of iterations and grid size. The algorithm spends between 76% ($128^3$ grid) to 82% ($512^3$ grid) of the time in level set evolution. This time can be reduced by adopting a more work efficient level set method that runs on the GPU [159].

Table 10: Performance Summary for Le Veque’s test

<table>
<thead>
<tr>
<th>Grid Size</th>
<th>Mean # Springls</th>
<th>Deformation Error (voxels)</th>
<th>Tracking Error (voxels)</th>
<th>Total Iterations</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$128^3$</td>
<td>5K</td>
<td>0.38±0.28</td>
<td>0.93±1.07</td>
<td>500</td>
<td>35 sec</td>
</tr>
<tr>
<td>$256^3$</td>
<td>19K</td>
<td>0.34±0.31</td>
<td>0.92±1.05</td>
<td>1000</td>
<td>300 sec</td>
</tr>
<tr>
<td>$384^3$</td>
<td>41K</td>
<td>0.31±0.27</td>
<td>0.91±1.08</td>
<td>1500</td>
<td>970 sec</td>
</tr>
<tr>
<td>$512^3$</td>
<td>72K</td>
<td>0.32±0.30</td>
<td>0.94±1.18</td>
<td>2000</td>
<td>2500 sec</td>
</tr>
</tbody>
</table>
Figure 50: Enright Test for grid size $384^3$ showing time $t=1.5$ for period $T=3$. (a) Springls constellation and (b) iso-surface.

Figure 51: Computation time per iteration as a function of (a) iteration and (b) grid dimension for LeVeque's test.

We also repeated Zalesak's test at a grid size of $128^3$ to compared it against implementations of Particle Level Sets (PLS) [201] and Marker Level Sets (MarkerLS) [199] (Table 11). Finally, we compared SpringLS to MarkerLS for LeVeque's test ($T = 2.5$ and grid size $128^3$) based on published results. The volume change for MarkerLS was 1.8% and SpringLS was 4.6%. However, the MarkerLS implementation [199] does not re-sample and uses 98K particles for LeVeque's
test. This is compared to SpringLS that uses only 5K springls on average and does re-sample. If we increase the grid size to 256\(^3\) for SpringLS, 19K springls are used on average and the volume change decreases to 1.2%.

**Table 11: Performance on Zalesak’s test for grid size 128\(^3\).**

<table>
<thead>
<tr>
<th>Method</th>
<th>Re-sampling</th>
<th>Device</th>
<th>Speed</th>
<th>Volume Change</th>
<th>Mean # Particles</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpringLS</td>
<td>Yes</td>
<td>CPU</td>
<td>20 fps</td>
<td>0.6%</td>
<td>2.8K</td>
</tr>
<tr>
<td>PLS</td>
<td>Yes</td>
<td>CPU</td>
<td>2.5 fps</td>
<td>2.6%</td>
<td>80K</td>
</tr>
<tr>
<td>PLS</td>
<td>No</td>
<td>CPU</td>
<td>3 fps</td>
<td>0.3%</td>
<td>86K</td>
</tr>
<tr>
<td>MarkerLS</td>
<td>No</td>
<td>GPU</td>
<td>28 fps</td>
<td>0.9%</td>
<td>49K</td>
</tr>
</tbody>
</table>

The Particle Level Set library (PLS) [201] uses 1 core. SpringLS uses 8 cores. MarkerLS from Mei et al. [199] does not re-sample.

### 5.7.4 Tracking Performance

Tracking performance is important for imaging applications, but there has been little quantitative reporting of it in literature for re-meshing and re-sampling schemes. Figure 52 shows SpringLS tracking and deformation performance on Enright’s tests at a grid resolution of 256 \(\times\) 256 \(\times\) 256. Deformation and tracking errors were measured for Zalesak’s test at the end of each revolution, and they were measured for LeVeque’s test after a full deformation period. Errors for Zalesak’s test are sub-voxel, and errors for LeVeque’s test increase linearly with the amount of deformation up until \(T = 3\); at which point, the object tears during deformation and error increases significantly.

Finally, we compared SpringLS with implicit level set tracking [115]. SpringLS performs slight better (Table 12), but the main advantage, which we will demonstrate later, is that SpringLS can circumvent the CFL time step restriction for methods that require large step sizes.

One limitation of SpringLS is that the density of springls is tied to the resolution of the level set through Marching Cubes in the FillGaps step, whereas representations that use particles can always increase the density of particle sampling to improve accuracy, regardless of grid resolution. LeVeque’s test used 5K springls on average at a grid size of 128\(^3\) whereas the Marker Level Set method used 98K particles at a grid size of 128\(^3\) [199]. More particles translates into more degrees of freedom, which implies lower material viscosity. For this reason, SpringLS are
not as good at modeling materials with low viscosity or for problems that require accuracy well below the grid resolution. This is less a concern for imaging applications where image resolution, contrast, and noise impose limits on segmentation accuracy.

![Figure 52: Deformation and tracking errors for Zalesak’s and Le Veque’s test. Distances are measured in voxels.](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Grid Size</th>
<th>Mean Deformation Error (voxels)</th>
<th>Mean Tracking Error (voxels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSID</td>
<td>100³</td>
<td>0.34</td>
<td>1.7</td>
</tr>
<tr>
<td>LSPC</td>
<td>100³</td>
<td>0.34</td>
<td>0.63</td>
</tr>
<tr>
<td><strong>SpringLS</strong></td>
<td>100³</td>
<td><strong>0.26</strong></td>
<td><strong>0.57</strong></td>
</tr>
<tr>
<td>LSID</td>
<td>200³</td>
<td>0.38</td>
<td>1.4</td>
</tr>
<tr>
<td>LSPC</td>
<td>200³</td>
<td>0.38</td>
<td>0.68</td>
</tr>
<tr>
<td><strong>SpringLS</strong></td>
<td>200³</td>
<td><strong>0.25</strong></td>
<td><strong>0.59</strong></td>
</tr>
</tbody>
</table>

Comparison with published results [115] for Level Sets with Interfacial Data (LSID) and Level Sets with a Point Correspondence (LSPC).

### 5.7.5 Topology-Preserving Deformations

Spring Level Sets can rigorously preserve topology through digital connectivity rules applied to the level set evolution scheme [183]. The rules detect when a modification to the level set at a particular voxel could change the object topology; in which case, the level set change is not permitted. To demonstrate this ability, Le Veque’s test was repeated with the topology-
preservation constraint. As seen in Figure 53, the object’s topology was preserved under extreme deformations even though the constellation has no explicit connectivity.

![Figure 53: Le Veque’s test with topology-preservation constraint for period T=3. The thickness of the disk is not allowed to be below 0.01 voxels in regions where the topology constraint is enforced. This constraint results in the divot visible at t=0.8T.](image)

5.7.6 Constructive Solid Geometry

Spring Level Sets can be used for Constructive Solid Geometry (CSG) by performing Boolean operations on the signed level set representation of two objects. Springls far from the zero iso-level are then eliminated with **Contract**. SpringLS does not require the intersecting region between objects to be re-meshed, which is necessary for mesh-based CSG. Figure 54 depicts the subtraction, intersection, and union of a cube and torus.

![Figure 54: (a) Subtraction, (b) intersection, and (c) union of a cube and torus. Each image shows springls on the left-side and the iso-surface on the right-side.](image)
Algorithm 5: Constructive Solid Geometry

```
foreach voxel X do
  //Perform boolean operation on level set functions
  \( \varphi_C(X) = \text{Boolean}(\varphi_A(X), \varphi_B(X)) \)
  Contract
  for \( z = 1:4 \) do
    Evolve
    Rebuild
```

5.8 Use Cases

5.8.1 Image Segmentation

Spring Level Sets was applied to active contour image segmentation [127] of objects driven under pressure forces from image intensities \( I(x) \) to find a specified iso-level \( \pi \):

\[
\frac{\partial p_n(t)}{\partial t} = \lambda_n \vec{n}_n(l(p_n(t)) - \pi).
\]  \hspace{1cm} (5.29)

It was implemented as a mixture of Java and OpenCL for the CPU. Parameter settings and grid size \((256 \times 256 \times 256)\) were fixed for all experiments. Experiments not initialized with an atlas were repeated with 4 different initializations (sphere, cube, torus, and metasphere [202]). Reported segmentation errors are measured as the average minimum distance from mesh vertices on the segmented mesh to iso-surface produced with standard level set segmentation.

Figure 55 shows segmentation of a pelvis from a CT image and highlights SpringLS’ ability to change topology\(^5\). Initializing the segmentation process with a cube or torus causes segmentation of the pelvis to include the femurs and spine as well. To mitigate this problem, we incorporate an atlas based approach.

---

\(^5\) We thank Dr. Ted Deweese and Dr. Lee Myers for their assistance in providing us with the data.
The segmentation result in Figure 55 can be improved by combining level set techniques with a parametric atlas. A Point Distribution Model (PDM) [37] statistical atlas of the pelvis was constructed with the standard PCA method [37] (Figure 56). Analogous statistical atlas methods have been developed for level sets [203], but these representations are not equivalent. The atlas was registered (rigid + global scale) to the CT image. The first 10 mode weights were optimized in increasing order to reduce the average distance from the atlas to target iso-surface in CT. Because the initial registration was fairly good (1.56±1.36 mm), optimizing the mode weights modestly improved the segmentation result to 1.40±1.32 mm. The registered mesh was then treated as a constellation of springs and advected towards the target iso-level \( \pi \) with an external velocity field \( \vec{v} \) produced by Gradient Vector Flow (GVF) [204] and pressure forces:

\[
\frac{\partial p_n(t)}{\partial t} = \lambda_p \vec{n}_n(1(p_n(t)) - \pi) + \lambda_r \vec{v}(p_n(t)).
\]  

(5.30)

This atlas based method produces a better pelvis segmentation (Figure 56 and Figure 57) than the non-atlas approach and provides a mapping from each springl back to the atlas, enabling transfer of region labels on the atlas to the segmented pelvis. This technique is also significantly faster than the non-atlas based approach. Table 13 summarizes results from these experiments. Segmentation results are very similar to performing the same task with the Level Set Method (LSM). The average distance between the SpringLS and LSM segmentations for the non-atlas approach was 0.13±0.15 mm and atlas approach was 0.25±0.32 mm.

Table 13: Summary of image segmentation results. Surface distance measurements are relative to segmentation with level set method.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Iterations</th>
<th>Mean # Springls</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pelvis</td>
<td>400</td>
<td>49K</td>
<td>220-240 sec</td>
</tr>
<tr>
<td>Pelvis \w atlas</td>
<td>100</td>
<td>25K</td>
<td>30 sec</td>
</tr>
</tbody>
</table>
Figure 55: Non-atlas based pelvis segmentation when initialized with a (a) cube and (b) torus. We thank Dr. Ted Deweese and Dr. Lee Myers for their assistance in providing us with the data.

Figure 56: (a) Mean atlas shape and (b) segment pelvis after active contour segmentation with SpringLS when initialized with PDM. We thank Sharmishtaa Seshamani and Gouthami Chintalapani for providing the statistical atlas in (a).

Figure 57: Atlas based segmentation showing (a) registered PDM and (b) final active contour segmentation with SpringLS. We thank Sharmishtaa Seshamani and Gouthami Chintalapani for providing the statistical atlas in (a).
### 5.8.2 Tracking and Atlasing 4D CT

To demonstrate elastic and fluid behaviors of Spring Level Sets, we applied SpringLS to segmentation and tracking of objects in 4D CT (a time series of 3D CT images). The following imaging pipeline combines segmentation, registration, and atlasing into a single system that uses SpringLS as its only deformable model representation. In lieu of a statistical atlas, or to construct one, we first manually segment objects of interest in one CT image. In this case, the epicardium. The coarse manual segmentation is then improved with active contour segmentation methods (Figure 58a). CT images at all other times are non-rigidly registered [205] to the reference time frame. The registration produces a displacement vector field representing an elastic deformation of the source image into the reference image. To track objects, the displacement field is applied to springls followed by **UnsignedToSigned, Evolve, and Rebuild** (Figure 58b). The result is a 1-to-1 mapping of springls found in the reference time frame to all other time frames. Because the mapping is 1-to-1, the rigid component of motion can be factored out and the non-rigid component analyzed with PCA to construct the statistical atlas shown in Figure 59.

![Figure 58: (a) Manual segmentation of one time frame shown in green and active contour segmentation shown in pink. After active contour segmentation, the epicardium is tracked to all other time frames. (b) shows one time frame of tracked epicardium. We thank Dr. Todd McNutt for providing data.](image-url)
This 4D tracking pipeline has only one transition between level set and mesh techniques. To further motivate the need for Spring Level Sets, we apply them to a problem that requires repeated transitioning between level set and mesh techniques.

### 5.8.3 Morphing Animated Characters

There is commercial interest in animated characters that undergo large non-rigid deformations (morph) as they execute a kinematic motion \[206, 207\]. Morphing is useful for reconstruction of animated characters as well \[208, 209\]. To address this class of problems, we present the following pipeline for morphing animated characters (Figure 60) with SpringLS. The models are first auto-rigged with Pinnochio \[210\] to fit a kinematic skeleton and assign bone weights to the mesh. The source model is then non-rigidly rescaled to align its skeleton with the target model, and both models are converted into SpringLS. During each animation time step, both the source and target model are articulated with Linear Blend Skinning (LBS) \[211\] to match kinematics of the animation.

Linear Blend Skinning is a type of elastic deformation that is popular in real-time character animation. Associated with each mesh vertex is a vector of bone weights \(w_k\) indicating the contribution of each bone to the deformation of the vertex. The springl constellation is deformed by applying a linear combination of the skeleton’s rigid bone transforms \(A_k\) to triangle vertices \(q_{n,m}\), weighted by the bone weights (eq. 31).
\[
\frac{\partial q_{n,m}(t)}{\partial t} = \left( \sum_k w_k(a_n) A_k \right) q_{n,m}(t). \tag{5.31}
\]

Linear Blend Skinning (LBS) is followed by 10 active contour iterations to deform the source model into the target model via,

\[
\frac{\partial p_n(t)}{\partial t} = \lambda_p \overrightarrow{n}_n H_\varepsilon \left( \varphi_b(p_n(t)) \right), \tag{5.32}
\]

where \( \varphi_b(x) \) is the target model’s signed level set and function \( H_\varepsilon(x) = \frac{1}{\pi} \arctan \left( x / \varepsilon \right) \) where \( \varepsilon = 0.25 \). The process of alternating between “elastic” LBS deformation and “fluid” active contour deformation is repeated until the source model aligns with the target. Bone weights are tracked from source to target using springls. In order to apply kinematics, it is necessary to track bone weights during LBS deformation.

Spring Level Sets guarantee that at all times during the animation, the model is watertight, non-self-intersecting, and in point-to-point correspondence with the original model. This morphing technique is advantageous in that it doesn’t require point correspondences between source and target meshes; and in fact, could be used to establish point correspondences.
Figure 60: Morphing pipeline showing (a) input meshes, (b) auto-rigged skeletons, (c) mesh rescaling to align skeletons, (d) morphing of a walking humanoid character. SpringLS constellation for morphing character is shown overlaid on silhouette of target character. Meshes courtesy of Cosmic Blobs® (Dassault Systems SolidWorks Corp.).

5.9 Discussion

Spring Level Sets are distinguished from previous work by their treatment of meshes as a constellation of disconnected triangles. SpringLS can behave as either a mesh or level set to perform better on tasks which require behaviors of both (Table 14).

SpringLS are similar to surfels, for which there is work to support the claim that surfels are equally effective at modeling fluid and elastic deformations as other representations [212, 213]. SpringLS can be transformed into surfels using a springl's particle location to define a disk's center and gradient of the signed level to define the disk's normal (see Figure 4). However, there are reasons to choose springls over surfels. First, SpringLS can be interpreted as mesh or level
set, so they can interact naturally with other geometric models during collision detection and rendering; and secondly, there is no shape, topology, or tracking information lost in the transformation from mesh or level set into SpringLS. This is crucial for applications that alternate between mesh and level set deformations, as demonstrated by the character morphing example.

Shape preservation is affected by the choice of inter-springl forces. Although we have empirically demonstrated one choice of inter-springl forces perform well on numerous examples, there are other choices that may have desirable properties. One may also choose not to relax or resample a subset of springs to preserve sharp features and fine details.

SpringLS spends a large percentage of computation time evolving a level set to keep the model's implicit representation consistent with the model's parametric representation. This redundancy and overhead is not unique to SpringLS. Marker/particle level set methods [90, 174, 214] and some re-meshing schemes [171, 172, 174] also maintain an implicit representation to provide a fast inside/outside test and handle topology changes.

Table 14: Comparison of deformable model representations. Check marks indicate the preferred representation for each behavior.

<table>
<thead>
<tr>
<th>Model Behavior</th>
<th>Mesh</th>
<th>Level Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid Transform</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Elastic Deformation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Topology Preservation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Point Tracking</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Plastic Deformation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Topology Change</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Self-intersection Prevention</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Collision Handling</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

5.10 Limitations

There are limitations to the SpringLS representation. As previously mentioned, the size of surface elements is tied to the Marching Cubes algorithm in the FillGaps step. Up-sampling the number of surface elements requires up-sampling the grid, and the grid's resolution imposes a CFL restriction on step size for active contour deformation (Algorithm 3). Fortunately, SpringLS has a method for circumventing the CFL time step restriction (Algorithm 4).
SpringLS only tracks points; and by doing so, sacrifices the continuity of the mapping between source and target shapes. Such sacrifice is necessary to address deformations where the object changes topology and the continuity of the mapping is disrupted. Moreover, new point correspondences are interpolated with neighbors in world space as opposed to neighbors in a geodesic neighborhood on the manifold. There is the potential to make large mapping errors for shapes with thin or folded geometries where these neighborhoods differ drastically. Surfels and SpringLS share these challenges, and we refer the reader to literature on surfels for possible solutions [91].

Relaxation and re-sampling processes modify the distribution and size of triangles in the original mesh. The interplay between relaxation and re-sampling is not well understood. We can only provide empirical evidence at this time that the procedure appears to be stable. Stability is indicated by convergence of the number of added / removed springls. In parameter tuning experiments, parameter choices that minimized the total number of added / removed springls correlated with convergent behavior (Figure 49). As an example, we executed the active contour algorithm with no advection forces and plotted the number of added and removed springls (Figure 61) for the Igea model (Figure 9). It is encouraging to see that the number of added / removed springls decreases quickly and falls below 1%. We would like to investigate why it does not go to zero in future work.

![Stationary Igea Model](image)

Figure 61: Number of re-sampled springls for the Igea model (Figure 9). Percentages are measured relative to the initial number of springls.
Contributions:

- The Spring Level Set data structure merges triangle meshes and level sets into one representation.
- We have described how to couple a level set with deforming mesh.
- We have explained how to re-sample a constellation moving triangles.
- An active contour segmentation algorithm has been described for SpringLS.
- An elastic deformation algorithm has been described for SpringLS.
- A parallel implementation of the algorithm has been described.
- We have described how to perform Boolean operations on SpringLS shape models.
- A pipeline for morphing animated characters has been described.
- We have described how to generate shape instances with PCA that preserve geometric properties.
Chapter 6

6 Multi-Object Spring Level Sets (MUSCLE)

6.1 Motivation

The previous two chapters introduced new data structures to address drawbacks of level sets, meshes, and label masks. In this chapter, MOGAC and SpringLS are combined to create a multi-object version of SpringLS. The need for such a data structure is motivated by image pipelines that combine registration and segmentation. The following section reviews work on combined registration and segmentation, the data structures that have been used to implement such pipelines, and the drawbacks of existing data structures for this application.

6.2 Background

6.2.1 Combined Registration and Segmentation

Deformable registration [6, 7] has become a popular technique for reliably and automatically segmenting multiple objects with little prior knowledge of anatomy or imaging technology. The output of the algorithm is a displacement field describing where voxel locations in the target image map to in the source image. This 3D-to-3D mapping allows any geometric structures identified in the source image to be warped into the target image to create a segmentation of the target [215]. To overcome the registration algorithm's lack of prior knowledge, there is a strong assumption that the target's anatomy is a smooth elastic deformation of the source's anatomy. This is rarely the case, leading to what is known as "atlas bias" where the warped source image still resembles the source image, possibly more than the target. One way to reduce bias is to repeat the registration with different source images and combine segmentations afterwards to create an "average" [216]. Methods that combine multiple segmentations produce "average" segmentations that are smoother than any individual
segmentation [217]. Alternatively, one can reduce bias and increase the image segmentation’s fidelity by following elastic registration with “fluid-like” segmentation [59, 218]

6.2.2 Hybrid Warping

The Hybrid Warping method [218] combines volumetric and surface-based registration to align cortical surfaces. Images are first registered with HAMMER [7]; then, the displacement field produced is used to warp a triangle mesh corresponding to a cortical surface in the source image. To accurately align the cortical surface with another cortical surface extracted from the target image, the warped triangle mesh is deformed with a customized surface warping technique that attempts to match regions of the cortical surface with similar curvature histograms. Because their system does not re-mesh, mesh quality can and will deteriorate, forming sharp edges, cusps, and creases. This is particularly detrimental to their surface registration algorithm that relies on stable curvature estimates to match mesh features.

6.2.3 Combined Volumetric and Surface Registration

Postelniciu et al. [59] use Combined Volumetric and Surface (CVS) registration to segment all cortical and sub-cortical structures in the brain. To do so, a tetrahedral mesh is constructed for the entire image domain. More tetrahedra are placed near the cortical surface so that the registration method will have greater precision in that region. A linear elastic system is solved on the tetrahedral mesh to deformably register source and target images with knowledge of corresponding features on the cortical surface. To insure a diffeomorphic mapping, tetrahedra must not flip during deformation nor can the domain be re-meshed. They use a tet smoothing scheme to fix the mesh topology, but the scheme is not guaranteed to eliminate inversions. In fact, there are no algorithms that are proven to not generate and/or remove inversions. The diffeomorphic constraint restricts the size of local deformations during deformable registration, which is unfortunately linked to the granularity and parameterization of the tet mesh.

Despite drawbacks of the CVS approach and the Hybrid Warping approach, coupling registration and segmentation does produce better results and reduces atlas bias from the registration phase. Since drawbacks of either approach are linked to difficulties with manipulating the
geometric data structure, we will present a new data structure that addresses these difficulties. Traditional data structures used to cascade registration and segmentation either lose information and/or make it difficult to preserve important geometric properties. We now review those data structures and the challenges each one poses.

6.2.4 Meshes

Triangle meshes have the ability to encode surface labels and track point correspondences. Displacement fields produced by registration can be directly applied to triangle meshes. Even though displacement fields represent smooth elastic deformations, the deformed triangle mesh can have sharp edges and cusps that were not present in the original mesh. To reduce artifacts, the mesh must be smoothed and re-sampled (re-meshed) to produce a higher quality mesh. Re-meshing is challenging and interferes with point correspondences in non-intuitive ways [46]. There is an intuition behind maintaining point correspondences with 1-to-4 subdivision of triangles, but rarefaction of correspondences with edge collapses [219] are less intuitive. Re-meshing becomes more difficult when deformations are viscoelastic or fluid-like, objects share boundaries or slide against each other, and practical geometric constraints are enforced. These constraints include that the mesh should not intersect adjacent structures or itself, and that it should not create air-gaps between adjacent structures.

6.2.5 Level Sets

Objects can be represented with sub-voxel precision as level sets [66]. Level sets are functions of 3D space that are stored as images. A common choice for level set function is the signed distance field with image intensities giving the minimum distance to the object's boundary. Voxel locations inside the object are negative and outside the object are positive. The object's boundary is then the iso-surface corresponding to the zero iso-level. Level sets naturally ensure that structures do not self-intersect, and it is simple to enforce that they do not overlap or have air-gaps [69]. The drawback to level sets is that they do not maintain point correspondences or surface labels, and application of a displacement field requires re-sampling the level set image, which acts as a low-pass filter by smoothing underlying geometric structures.
6.2.6 Label Masks

It is common for objects to be manually segmented with a painting tool to produce binary masks. These masks are then merged together to create a label mask image. Displacement fields can be applied to labels by warping and re-sampling the label mask image. Re-sampling acts as a low-pass filter that smoothes underlying structures; and because label image values are region indicators, only nearest-neighbor interpolation can be used, resulting unwanted aliasing artifacts. Label masks have only voxel precision and have no ability to maintain point correspondences or surface labels; however, they are very popular because they can be stored and manipulated with a small memory footprint.

6.3 Overview

In the previous chapter, we described how to couple a constellation of disconnected triangular surface elements with a level set through Spring Level Sets (SpringLS). The constellation is capable of tracking point correspondences and surface labels, and the level set enforces geometric constraints. This chapter extends SpringLS to the multi-object case and shows how to use it in the context of global registration, deformable registration, and segmentation. The new data structure is a combination of three existing data structures: a constellation of springs, a label mask, and a distance field. The Multi-Object Spring Level Set (MUSCLE) data structure is simultaneously a triangle mesh, level set, and label mask and addresses the drawbacks of each. To demonstrate, we cascade diffeomorphic Demons [13] registration and active contour segmentation [66] to segment and label 74 structures in MR images of the human brain.

6.4 Representation

The MUSCLE data structure is depicted in Figure 62 and corresponding symbols are defined in Table 15. A springl $S_n$ is a triangular surface element consisting of a particle $p_n$ and three springs connecting the particle to each of the triangle’s vertices $q_{n,m}$ (Figure 62a). Each springl maintains an object label $l_n \in \mathcal{L} = \{1, \cdots, L\}$ and a correspondence point $\alpha_n$ that maps the particle $p_n$ to a point on the original model. Each object has its own springls constellation that
encloses the object’s interior. Constellations for all objects (Figure 62b) are merged into one constellation for storage and manipulation. The label mask \( \chi: \Omega \rightarrow \mathcal{L} \) (Figure 62c) maps each voxel in the image domain \( \Omega \subset \mathbb{R}^3 \) to an object label \( l \). The distance field \( \psi: \Omega \rightarrow \mathcal{R} \) (Figure 62d) measures the distance of each voxel to the nearest object represented by level sets \( \varphi_l: \Omega \rightarrow \mathcal{R} \) (i.e. \( \psi(x) = \min_l |\varphi_l(x)| \)). The signed distance \( \varphi_l \) for each object can be recovered at the boundary

\[
\Lambda_l = \{ x | \exists y \in \mathcal{N}(x) \text{ s.t. } \sigma_l(x) \neq \sigma_l(y) \} \tag{6.1}
\]

of each region \( l \), where \( \mathcal{N}(x) \) is the 6-connected neighborhood of voxel \( x \) and the sign \( \sigma_l(x) \) is indicated by,

\[
\sigma_l(x) = \begin{cases} 
-1 & \chi(x) = l \\
1 & \text{otherwise}
\end{cases}
\tag{6.2}
\]

Table 15: Definition of symbols for MUSCLE.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p _n )</td>
<td>Particle location.</td>
</tr>
<tr>
<td>( a _n )</td>
<td>Corresponding particle location on initial surface.</td>
</tr>
<tr>
<td>( t _{n,m} )</td>
<td>Triangle vertex ( m = {0,1,2} ).</td>
</tr>
<tr>
<td>( \vec{n} _n )</td>
<td>Triangle normal.</td>
</tr>
<tr>
<td>( S _n )</td>
<td>4x4 matrix describing a springl.</td>
</tr>
<tr>
<td>( l _n \in \mathcal{L} = {1, \ldots, L} )</td>
<td>Label for springl ( n ).</td>
</tr>
<tr>
<td>( \varphi _l(X) )</td>
<td>Level set for object with label ( l ).</td>
</tr>
<tr>
<td>( \tilde{\varphi} _l(X) )</td>
<td>Partially reconstructed level set for object label ( l ).</td>
</tr>
<tr>
<td>( \Lambda_l )</td>
<td>Boundary voxels for object label ( l ).</td>
</tr>
<tr>
<td>( \sigma_l(X) )</td>
<td>Level set sign for object label ( l ).</td>
</tr>
<tr>
<td>( \chi(X) )</td>
<td>Label image.</td>
</tr>
<tr>
<td>( \psi(X) )</td>
<td>Distance field.</td>
</tr>
<tr>
<td>( \omega(X) )</td>
<td>Unsigned level set for springl constellation.</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>Smoothing weight.</td>
</tr>
<tr>
<td>( \rho_l(X) )</td>
<td>Outward pressure for object label ( l ).</td>
</tr>
<tr>
<td>( \vec{\sigma}(X) )</td>
<td>External velocity field.</td>
</tr>
</tbody>
</table>
Figure 62: (a) Diagram of springl. (b) Springl constellation for two objects. (c) label mask. (d) clamped distance field. (e) Raycast rendering of MUSCLE data structure.

We use the convention that distance measurements at locations inside the object have negative values and outside have positive values. The partially reconstructed level set \( \tilde{\phi}_l : \Lambda_l \rightarrow \mathcal{R} \) is given by \( \tilde{\phi}_l(x) = \sigma_l(x) \psi(x) \). \( \tilde{\phi}_l \) measures the signed distance at the boundary (\( \Lambda_l \)) between objects and provides enough information to extract an iso-surface with marching cubes [162] or recover the entire signed distance field for each object with fast-marching [157]. More importantly, the "label mask + distance field" data structure avoids having to store and manipulate independent level set images for each object. Additionally, having a unified representation guarantees consistency between all three data structures such that there are no self-intersections, air-gaps, or overlaps. For an \( M \times M \times M \) image containing \( L \) objects, the amount of memory needed to store the MUSCLE data structure is \( O(M^3 + LM^2) \). This is compared to \( O(LM^2) \) for meshes, \( O(LM^3) \) for independent level sets, and \( O(M^3) \) for label masks. Equipped with the MUSCLE data structure, we proceed to discuss its usage in the context of registration and active contour segmentation.

6.5 Global Registration

Deformable registration is sensitive to the initial alignment of source and target images. To obtain a good initialization, an image-based global registration algorithm such as FLIRT [5] can be used to estimate an affine transformation between source and template images. The \( 4 \times 4 \) transformation matrix \( A \) is then applied to the springls constellation (Figure 63a), labels mask, and distance field. Nearest-neighbor interpolation is used to transform the label mask, and
trilinear interpolation is used for the distance field. The interpolation filters both the labels and
distance field, producing iso-surfaces not quite as smooth as the originals (Figure 63b). However,
transformations can be applied to a constellation of springs without interpolation error. We
therefore use the constellation to correct errors in the label mask and distance field. To do so,
level sets representing each object are evolved to minimize their distance to the constellation.
This is accomplished by first constructing the clamped distance field for all springs:

\[ \omega(x) = \min\{d_{max}, d_1(x), \ldots, d_N(x)\} \]  \hspace{2cm} (6.3)

where \( d_n(x) \) is the distance from location \( x \) to spring \( n \), and \( d_{max} = 0.5 \) is the clamped
distance. Level sets \( \phi_i \) are evolved to minimize the following objective function:

\[ E = \sum_i \int \left( \frac{1}{2} \omega(x)^2 + \lambda |\nabla \phi_i(x)| \right) \delta(\phi_i(x)) dx \]  \hspace{2cm} (6.4)

where \( \lambda \) is a regularization weight that controls the model's smoothness. Solving the Euler-
Lagrange equations, eq. 4 can be minimized with the following scheme:

\[ \phi_i^{t+1}(x) = \phi_i^t(x) - \Delta t \delta_e(\phi_i^t(x)) \left( \omega(x) \nabla \omega(x) \cdot \frac{\nabla \phi_i^t(x)}{|\nabla \phi_i^t(x)|} + \lambda \nabla \cdot \frac{\nabla \phi_i^t(x)}{|\nabla \phi_i^t(x)|} \right) \]  \hspace{2cm} (6.5)

where \( \delta_e(d) \) is a compactly supported approximation to the dirac delta [66]. The iterative
scheme is implemented with Multi-Object Geodesic Active Contours (MOGAC) [220] (Chapter 4)
because it uses the "label mask + distance field" data structure and does not create air-gaps or
overlaps between adjacent structures. Figure 63c illustrates that this method is effective at
boosting the fidelity of level sets that undergo global registration.
Figure 63: Cortical surface after applying Affine transformation to (a) triangle mesh, (b) label mask and distance field (c) MUSCLE after 20 iterations of eq. 5 with $\lambda = 1$.

### 6.6 Deformable Registration

Once source and target images have been roughly aligned with global registration, they are more precisely aligned with deformable registration. Image based registration algorithms output a displacement field $\tilde{\mathbf{v}}_{ts}: \Omega \mapsto \mathbb{R}^3$ describing where each voxel in the target image $I_t: \Omega \mapsto \mathbb{R}$ maps in the source image $I_s: \Omega \mapsto \mathbb{R}$. The source image, label mask, or distance field, can be transformed into the target via $I_t(x) \mapsto I_s(x - \tilde{\mathbf{v}}_{ts}(x))$. Displacements fields representing the forward mapping $\tilde{\mathbf{v}}_{st}: \Omega \mapsto \mathbb{R}^3$ can be applied to mesh vertices $q_{n,m}$ in the springls constellation to obtain new positions $\dot{q}_{n,m}$ via $\dot{q}_{n,m} = q_{n,m} + \tilde{\mathbf{v}}_{st}(q_{n,m})$. Unfortunately, most registration algorithms produce displacement fields that are not isomorphic (i.e. $\tilde{\mathbf{v}}_{st}(x - \tilde{\mathbf{v}}_{ts}(x)) \neq \tilde{\mathbf{v}}_{ts}(x)$).

A different procedure is needed for transforming label masks and distance fields so that their iso-surfaces are well aligned with springls constellations. To do so, springls are incrementally displaced along linear trajectories from source to target:

$$q_{n,m}(t) = q_{n,m}(0) + t\tilde{\mathbf{v}}_{st}(q_{n,m}(0))$$

(6.6)

where $t \in [0,1]$. After the $k$ iteration, corresponding to time $t = k\Delta t$ for $k = 0, 1, ..., [1/\Delta t]$, the label mask and distance field are evolved to track the moving mesh via eq. 5. The step size is chosen to be $\Delta t \leq d_{max}/\max_{n,m}\|\tilde{\mathbf{v}}_{st}(q_{n,m})\|$. The iterative scheme in eq. 5 is repeated 4 times per iteration of eq. 6. Figure 64a illustrates that applying a displacement field to a mesh can create thin cusp structures that self-intersect. In the MUSCLE framework, the level set
representation that is evolved with the mesh cannot develop self-intersections, but it can change topology (Figure 64b). This can be addressed by using the simple-point test [183] to preserve the object’s topology (Figure 64c).

![Figure 64: Cortical surface after applying displacement field to (a) triangle mesh, (b) MUSCLE (c) MUSCLE with topology-preservation constraint. Back-face of the surface is shown in red, indicating self-intersection. Red circle indicates region with topology change.](image)

### 6.7 Active Contour Segmentation

After registration, active contour methods are used to better align the model with anatomical boundaries visible in the target image. An active contour segmentation framework was presented in the previous chapter for evolving Spring Level Sets. We extend the advection phase to multiple objects by considering advection equations of the following form,

\[
\frac{dp_n}{dt} = \lambda_p \left( \rho_{i_n}(p_n) - \max_{i \neq i_n} \rho_i(p_n) \right) \vec{n}_n + \lambda_\sigma \vec{v}(p_n),
\]

where \( \rho_i: \Omega \rightarrow \mathcal{R} \) is the pressure for object \( i \), \( \vec{v}: \Omega \rightarrow \mathcal{R}^3 \) is an external velocity field, and \( \vec{n}_n \) is the outward pointing normal for spring \( n \). The first term moves the boundary to classify location \( p_n \) as either inside or outside object \( i_n \) via the pressure difference between \( i_n \) and the maximum pressure from objects in the neighborhood \( \mathcal{N}(p_n) \). This is analogous to the outward normal force used in the SpringLS to drive the object’s boundary towards an iso-level in an image; except now, objects that share boundaries with other objects compete by pushing on each other via their normal force. The second term moves boundaries for all objects towards edges in the image. The terms \( \lambda_p \setminus \lambda_\sigma \) control the relative contributions of each force to the movement of a spring.
SpringLS are incrementally moved with eq. 7. The time step is chosen so that no springl moves more than $d_{max}$ during the advection phase. The relaxation phase uses inter-springl attraction forces to smooth and regularize the constellation. The only difference between the single and multi-object version of SpringLS is that springls are only permitted to interact with springls that have the same label $l$ during relaxation and re-sampling (See Section 5.5 for details). This is equivalent to having separate spring level sets for each object. Likewise, the re-sampling phase removes springls with label $l$ that are too far from $\hat{\phi}_l$'s zero iso-level or are too small. This includes removal of sharp edges and cusp structures (Figure 64a) that lie far from any level set's zero iso-level.

As before, springls that are too large are split in half and large holes in the constellation are filled by adding springls. Point correspondences for new springls are interpolated based on point correspondences for neighboring springls. The difference is that MUSCLE fills holes in the constellation with respect to each object individually as opposed to the union of all springls.

In the level set evolution phase, the label mask and distance field are evolved to track the constellation via eqs. 3-5. The active contour deformation follows the same procedure outlined in the previous chapter (Algorithm 1). Results for active contour segmentation with Multi-Object SpringLS will be presented in the next chapter.

### 6.8 Atlas Construction

A point distribution model can be constructed for a collection of MUSCLEs with the standard PCA method [37] as long as re-sampling was disabled during deformation. This restriction limits MUSCLE to elastic deformations, but that is acceptable because point distribution models are intended to model shape variation with only tens to hundreds DOF. Once a PDM is constructed, instances of the constellation can be generated by choosing a set of mode weights. The shape deformation induced by changing mode weights will usually violate the CFL condition. Therefore, we use **UnsignedToSigned** (Section 5.5.4) to estimate the boundary of each object based on its constellation. The coarse estimate for each object's level set may overlap or enclose level sets for other objects. We use a greedy approach for rasterizing and merging multiple level sets based on a pre-specified nesting order. The order in which level sets are rasterized should
start with the inner most objects and proceed outwards. The nesting order is determined automatically based on the template label mask. The computational complexity of greedy unsigned-to-signed conversion is $O(LM^3)$ for $L$ labels $M^3$ voxels, and the memory complexity is $O(M^3)$. Once level sets for all objects have been merged into a label mask + distance field, object boundaries are better localized through multi-object level set evolution (eqs. 3-5). The resulting MUSCLE representation satisfies many important geometric properties (no self-intersections, air-gaps, overlaps, etc ...).

In this application, unsigned-to-signed conversion is unavoidable because the mean constellation does not have a corresponding label mask and distance field, and PCA on label masks and distance fields does not produce the same result as PCA on triangle constellations. Unsigned-to-signed conversion can become a computational bottleneck if the number of PCA shape instances or the number of labels ($L$) is large. An alternate approach is to perform the unsigned-to-signed conversion only for the mean shape and then deform the mean into individual instances with the procedure for deformable registration (Section 5.6). Deformable registration is much faster ($O(M^2)$ time complexity) than unsigned-to-signed conversion for multiple objects. Methods that use PDMs for registration and segmentation only require a mesh representation, so generating a label mask + distance field can be postponed to the end of the pipeline.

6.9 Case Studies

6.9.1 Cardiac Tracking in 4D MRI

The following case studies demonstrate the utility and versatility of MUSCLE in real-world applications. Our first example examines cardiac tracking in 4D MRI. The human heart has multiple components that have different anatomical importance. It is difficult to geometrically model because it is a non-manifold structure. Modeling each component of the heart as a different triangle mesh will create small gaps and overlaps between components when the model undergoes deformation. The heart can be modeled as a collection of level sets, but level sets cannot model twisting motion and large elastic deformations which characterize the heart's true behavior. A common solution is to use a tetrahedral mesh to model each component of the
heart. High-quality tetrahedral models have been built that behave well under finite element simulation of beating motion \[221\]. However, tetrahedral meshes require linear elastic regularization to avoid tetrahedra flipping \[222\]. MUSCLE does not require regularization (at least not for this reason) or safeguards during deformation, and can be used to automatically adapt an existing cardiac model to an individual patient based on their 4D CT or MRI.

As an example, MUSCLE was used to improve the segmentation of a patient's heart that had been semi-automatically segmented into 4 structures (Figure 36). Meshes for each structure had small gaps and overlaps. MOGAC was used to remove these small gaps and overlaps so that the model is a proper partition of the epicardium. The model was then converted to MUSCLE, and the corresponding CT image was deformably registered to all other time frames with VABRA \[11\]. The MUSCLE model was then deformed to match the geometry of each time frame. Figure 65 shows the complete tracking pipeline.

Figure 65: 4D tracking pipeline used for cardiac tracking in MRI and organ tracking in CT. The input to the system are 3D images representing each time frame.
Figure 66 shows segmentation of the heart at different time frames. The figure shows cross-sections of the heart overlaid on cross-sections of the MRI, as well as the epicardial surface showing springs and object labels.

Based on these segmentations, a Point-Distribution Model was constructed for the heart. PCA modes for the heart's motion are shown in Figure 68. Notice that the left ventricle slides along the right ventricle in PCA instances of the model. One direction for future research would be to couple this 4D segmentation with a finite element simulation of the heart through embedded mesh techniques [63].
Figure 66: Heart tracking in 4D CT showing the heart at equally spaced time points in the cardiac cycle. We thank Dr. Terry Peters for providing data.
Figure 67: Heart tracking in 4D CT showing the heart at equally spaced time points in the cardiac cycle. We thank Dr. Terry Peters for providing data.
6.9.2 Multi-organ Tracking in 4D CT

Radiation therapy is used to treat oncology cases by administering targeted doses of radiation to cancerous tumors. The planning phase involves designing a strategy for treating the tumor with an X-ray beam aimed from different directions. The hope is that by varying the direction, most of the radiation dosage will accumulate in the cancerous region and little will accumulate in surrounding organs. Organs have different sensitivities to X-Ray radiation; and depending on the
organ and radiation dosage, could have near-fatal effects on the patient. To mitigate complications from radiation therapy, important organs are segmented in pre-treatment CT images (3D or 4D) and used to assess the safety of the treatment plan based on radiation therapy simulations. During therapy, the pre-treatment CT image along with planning information is registered to the patient. A known problem with pre-treatment CT is that the patient's organs shift between visits, and motion of the lung and heart during therapy effect the accuracy of the treatment.

An important image analysis problem is to use a 4D CT acquisition to create radiation therapy plans that account for breathing. It is usually too labor intensive to manually or semi-automatically segment multiple organs in each time frame of the 4D CT. Instead, organs are semi-automatically segmented in one time frame and propagated to all others via deformable registration [223]. MUSCLE allows segmentations to be propagated to all time frames with sub-voxel accuracy. These segmentations can be used to construct a parametric atlas of the patient's gated breathing cycle.

To demonstrate, a 4D CT dataset for an oncology patient was semi-automatically segmented with Pinnacle3\(^6\). The segmentation masks were then refined with active contour methods to localize boundaries with sub-pixel precision. Every time frame was deformably registered [11] to the segmented time frame, and a MUSCLE representation of 4 organs was propagated to each time frame (Figure 69) using the pipeline shown in Figure 65. A point-distribution model was constructed for the patient. As shown in Figure 70, instances of the point-distribution have the property that none of the organs overlap or have air-gaps. More importantly, the lungs are able to slide on the epicardium and liver as they would in normal breathing. The level set that is coupled with the label mask makes it easy to mask CT volumes and assess dosage during simulation. Although dosage can only be assessed on a per organ basis with this method, a natural extension would be to parcellate or over-segment each organ so that dosage can be assessed with greater regional localization, or embed the mesh in a coarse tetrahedral or hexahedral mesh. In the MUSCLE framework, more structures can be added to the

segmentation with minimal overhead (i.e. memory and computation scale with the total surface area of all objects). The next example will examine how to register and segment a structure that has been parcellated into many different structures.

Figure 69: Multi-organ tracking in 4D CT showing 5 time points in the respiratory cycle. We thank Dr. Todd McNutt for providing data.
Figure 70: Point Distribution Model for respiratory cycle shown in Figure 69.
6.9.3 Full-brain Parcellation in MRI

Geometry

An emerging application for brain parcellations is to use DTI fiber tracks to assess connectivity of gyral regions [224]. To do so, the brain cortex and all its cortical and subcortical structures must be segmented. Existing systems use either label masks, which only have voxel precision, or coarse tetrahedral meshes. Tetrahedral meshes require large elements to reduce the risk that a tetrahedron will invert when it undergoes deformation. The following imaging pipeline segments the brain into 74 non-overlapping structures with sub-voxel / sub-millimeter precision. A MUSCLE was constructed for the brain by extending gyral labels produced by FreeSurfer [225] with fast-marching [66] and combining them with sub-cortical labels and the level set for the central surface produced by CRUISE [226] (Figure 71).

Figure 71: (a) gyral labels (top), sub-cortical labels (bottom). (b) Sub-cortical + extended gyral labels (top), level set for central surface (bottom). (c) MUSCLE label mask for 74 structures (top), MUSCLE distance field (bottom). (d) Raycast rendering of MUSCLE parcellation.

It may seem unusual to parcellate the brain in this way because gyral structures are only present in the Gray Matter (GM). However, fiber tracks can only be reliably found in the White Matter (WM); so to establish connectivity, gyral regions must be extended into the WM. The central surface lies in the "middle" of the GM. We chose to use the central surface in our parcellation because it can be found more reliability [226], and the WM/GM and Pial surfaces have sharp cusps and deep folds that violate the uniform smoothness assumption that is inherent in the regularization of eq. 4. The smoothing assumption is very common and needed in order to eliminate sharp cusps and creases.
Results

MUSCLE brain parcellations were constructed for 10 subjects from the OASIS cross-sectional database [168]. Each parcellation was used to segment the other 9 subjects (90 experiments total) via the following pipeline (see also Figure 72):

1) Each skull-stripped MRI was affine registered to the target.
2) The affine registered image was deformably registered to the target with diffeomorphic Demons [13].
3) The displacement field and affine transformation were applied to the MUSCLE parcellation.
4) The MUSCLE parcellation was evolved with active contour methods to find the central surface produced by CRUISE.

Figure 72: Flow diagram depicting the cortical image segmentation pipeline.
To relocate the central surface, which we have already segmented with CRUISE, represented by level set $\psi_c : \Omega \mapsto \mathcal{R}$, we use $p_c(x) = H_\epsilon(\psi_c(x))$ where $H_\epsilon(x) = \text{atan}(x/\epsilon)$ with $\epsilon = 0.25$, $\lambda_p = 1$ (pressure weight in eq. 7) and $\lambda_v = 0$ (external velocity field weight in eq. 7). This choice of pressure forces moves object boundaries towards the zero iso-level of the central surface and has the effect that the boundary between adjacent structures will not move during segmentation because their forces will cancel. It is possible design forces that cause shared boundaries to move; however, it is assumed in this pipeline that boundaries between adjacent structures were found accurately in the registration phase, and the residual error is due to the central surface only. Figure 73 and Figure 74 shows a morphing example using reg+seg and Figure 75 juxtaposes the final result of reg+seg with ground truth. Table 16 compares the accuracy of several approaches. Labeling accuracy is measured with the extended Jaccard metric [59].

Table 16. Performance summary. Distance measurements are for cortical surface only. Accuracies for referenced works do not reflect the same parcellations, datasets, or sample sizes.

<table>
<thead>
<tr>
<th>Pipeline</th>
<th>Geometric Representation</th>
<th>Template to Subject</th>
<th>Subject to Template</th>
<th>Sub-cortical Label Acc.</th>
<th>Cortical Label Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUSCLE registration</td>
<td>MUSCLE</td>
<td>1.29±0.45 mm</td>
<td>1.05±0.36 mm</td>
<td>53.1±13.8%</td>
<td>80.1±2.6%</td>
</tr>
<tr>
<td>MUSCLE reg + seg</td>
<td>MUSCLE</td>
<td>0.40±0.05 mm</td>
<td>0.25±0.01 mm</td>
<td>52.8±13.0%</td>
<td>81.2±2.7%</td>
</tr>
<tr>
<td>HAMMER</td>
<td>Label mask</td>
<td>--</td>
<td>--</td>
<td>66.8±6.7%</td>
<td>36.6±6.0%</td>
</tr>
<tr>
<td>Hybrid Warp</td>
<td>Triangle mesh</td>
<td>0.45±0.05 mm</td>
<td>0.37±0.05 mm</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>CVS</td>
<td>Tet-mesh</td>
<td>1.5 to 2.5 mm</td>
<td>--</td>
<td>70.5±4.6%</td>
<td>54.4±10%</td>
</tr>
</tbody>
</table>
Figure 73: MUSCLE reg+seg showing deformable registration (top row) and active contour segmentation (bottom row). MR data was obtained from the OASIS database.

Figure 74: MUSCLE reg+seg showing deformable registration (top row) and active contour segmentation (bottom row). MR data was obtained from the OASIS database.
Figure 75: MUSCLE parcellations for (a) source, (b) source after applying displacement field (18 sec) and affine transform (7 sec), (c) target, and (d) source after reg+seg (21 min total, 1.1M - 1.4M springs, $256^3$ voxels, 100 active contour iterations). Computation times do not include FLIRT (10 min), Diffeomorphic Demons (1 hr), or CRUISE (2.5 hrs). Dual Intel E5630 (8 cores). MR data was obtained from the OASIS database.
Discussion

MUSCLE reg+seg and Hybrid Warp [218] have similar pipelines. MUSCLE augments Hybrid Warp by providing a data structure with which to represent full brain parcellations that can re-sample / re-mesh under geometric constraints (i.e. no self-intersections, air-gaps, or overlaps). Table 16 reports MUSCLE produced more accurate cortical surfaces than CVS [59], which uses a tetrahedral mesh (tet-mesh). Tet-meshes are difficult to use for registration because diffeomorphic deformations require safeguards to avoid tetrahedral inversion. MUSCLE places no restrictions on diffeomorphic deformations and has other useful geometric properties. For example, MUSCLE iso-surfaces are smooth, which is not always the case for meshes (Figure 64) and rarely the case for iso-surfaces extracted from label masks. It is important to have smooth surfaces (along with point correspondences) because cortical surface analysis is often sensitive to surface curvature [225]. MUSCLE associates gyral labels with springls instead of voxels, which partially explains why cortical labeling accuracy is higher for MUSCLE. In HAMMER and CVS, the sampling density of voxel-based gyral labels is affected by GM thickness. Regions where the GM is thin are likely to report inaccurate labelings because the Jaccard metric is sensitive to a labeled region's volume. MUSCLE evenly samples the cortical surface with springls so that gyral labeling is unaffected by GM thickness. MUSCLE produced less accurate subcortical segmentations than other works because it used Demons instead of HAMMER. Demons segmentation with a label mask has a subcortical accuracy of 54.3±13.6%. MUSCLE's slightly lower accuracy (Table 16) is likely due to smoothing (eq. 4). Subcortical structures have sharper features than folds in the central surface, so they should have different smoothing weights. To improve subcortical accuracy, one could use HAMMER for registration and disable smoothing in subcortical regions.

6.9.4 Whole-body Segmentation in CT

Full brain parcellation highlights many of MUSCLE's geometric properties. However, full brain parcellation has been explored for many years and there are several systems for automatically segmenting the brain [32, 226-228]. An emerging and arguably more difficult problem is to segment the whole human body as observed in CT images. Although whole body segmentation
is a multi-object problem much like the brain, there are several additional challenges that are unique to the whole human body:

- **More anatomical variability:** Height and weight can vary significantly in clinical populations. Much of this variability manifests in different shapes and positions of bones and adipose (fat).

- **Image size:** A 0.5 mm resolution CT image is approximately 512 x 512 x 1024, which is sixteen times larger than standard brain MR images which are 256 x 256 x 256. Assuming linear complexity, it is expected that segmentation of the whole body will take sixteen times more memory and computation because not only is the problem size 8 times larger, but the CFL restriction requires the time step to be twice as small.

- **Soft-tissue contrast:** CT images have poor soft-tissue contrast compared to MRI, which makes locating boundaries between internal organs more difficult.

- **Truncation:** Most clinical full-body CT images are truncated at the arms, waist, and neck. Mismatches in the field of view between subjects violates the diffeomorphic assumption that underlies many deformable registration algorithms.

- **Material properties:** Material properties for the whole human body vary from rigid (bones), multi-rigid (joints), elastic (lungs), to fluid (blood). This is compared to the human brain which is largely homogeneous.

- **Sliding surfaces:** There are many surfaces within the body that have the ability to slide on each other. As mentioned in the introduction, sliding surfaces are difficult to model with triangle meshes, tetrahedral meshes, and level sets. This intra-subject variability is not present in the brain.

A system for whole body segmentation is beyond the scope of this work; however, we believe that MUSCLE addresses some of the challenges associated with whole body segmentation. In particular, MUSCLE can model a range of materials with different physical properties, and it addresses the sliding surface problem which is not well addressed by either mesh or level set techniques.

To demonstrate the potential to use MUSCLE for whole body segmentation, we constructed a segmentation pipeline for the whole body. The pipeline uses a template of the human body
called the XCAT [229, 230] that consists of hundreds of segmented structures. The XCAT phantom is based on manual segmentations of the Visible Human [231]. Internal organs have been rescaled to match demographics for the 50 percentile male and female. A CT machine simulator was used to synthesize realistic looking CT images of the XCAT phantom. The combination of the XCAT segmentation masks and corresponding synthetic CT allow whole body CT scans of individual subjects to be registered to the XCAT. After which, all template structures can be deformed to match the patient's whole body CT (see Figure 76). A point-distribution model was constructed based on 20 whole body segmentations (Figure 77).

Figure 76: Pipeline for constructing a Point Distribution Model from whole-body CT images and generating instances of the model.
Figure 77: Point Distribution model for 50 structures in the human body showing deviation from the mean along the first (largest) PCA mode. Cut-away of body (top row) and internal organs (bottom row). We thank Dr. Ben Tsui and Dr. George Fung for providing the XCAT and Dr. John Carrino for providing data.

MUSCLE guarantees that all instances of the model generated from the atlas contain no overlaps or air-gaps between adjacent structures. Instances of the atlas characterize some variation among the population, but we observed that individual segmentations based on deformable registration are not very accurate. There is simply too much variation between individuals to model inter-subject variability with an elastic deformation model. A system for coupling elastic registration with segmentation is under development and will hopefully address the challenges of whole body segmentation. New atlas techniques must also be developed to accommodate variation that is specific to the human form. For example, there has been successful work on augmenting point-distribution models to produce better atlases for full-body surface reconstructions [232, 233].
Contributions:

- Multi-object Spring Level Sets (MUSCLE) merge level sets, label masks, and triangle meshes into one data structure.
- We have described how to combine SpringLS and MOGAC to represent and track multiple objects.
- Methods have been described for using MUSCLE for global registration, deformable registration, active contour segmentation, and atlas construction.
Chapter 7

7 Parametric Images (p-images)

7.1 Motivation

So far, we have shown how to use MUSCLE with existing algorithms to provide interoperability between image registration, segmentation, and atlasing tasks. Although there is utility in providing this interoperability, we will go one step further in this chapter and describe a new image analysis method for constructing atlases. The method leverages MUSCLE to provide guarantees on geometric quality (e.g. no air-gaps, overlaps, or self-intersections). This chapter will connect back to Chapter 3 where we discussed how to simulate medical images based on level set representations of objects. Similar assumptions will be made regarding the homogeneity of image intensities within objects; but instead of modeling only a single object (i.e. bone cement), we will model multiple objects in the human body simultaneously.

7.2 Introduction

Deformable registration is commonly used to construct atlases of the human body from medical images. During registration, warping images with deformation fields can stretch or compress edges; and after registering images to a template, misalignment of anatomical boundaries and/or stretching of edges between anatomical boundaries will produce a blurry image when images are averaged together. This work introduces a novel parametric image (p-image) representation that preserves edge strength when averaging or deforming images, regardless of the registration algorithm's performance. The key idea is to decouple shape and intensity information, manipulate them independently, and then synthesize images. We will describe how to warp and form linear combinations of parametric images. Applications to whole brain and whole body atlas construction will be presented, but the method is not particular to these applications nor does it require any specific registration algorithm.
7.3 Background

One way to construct an anatomical atlas is to combine multiple 3D medical images (e.g. CT or MR) of the human body. The atlas should ideally capture variations in geometry and intensity for different tissue types. A common approach is to use deformable registration to create a dense 3D-to-3D mapping between template and subject [6, 7]. The template image can be deformed into the subject's image space or vice versa to construct an atlas.

One way to construct an image intensity atlas is to register all subjects to a template image and then average those warped subject images [7, 12]. A known problem with this approach is that the average image is blurry and has worse contrast than any individual image. In fact, the sharpness of the average image is commonly used to assess the performance of deformable registration algorithms [7]. Blurriness is an indicator of misaligned anatomical boundaries. If boundaries are misaligned, then the average intensity at a boundary voxel is computed across different structures, which creates ambiguity as to which structure the average intensity corresponds.

The atlas construction method just described characterizes differences in intensity, not geometry. To capture variations in geometry, the template image is deformably registered to all other subjects. After which, any geometric structures identified in the template image can be warped into the subject's image space. There are several approaches to analyze geometric differences. One approach is to create mesh segmentations of each anatomical structure, warp them into each subject's image space, and analyze them with Principal Component Analysis (PCA) [12]. Alternatively, one can transform image segmentations into a LogOdds representation and analyze them with PCA [234].

There have been several methods proposed for modeling variations in shape and intensity information simultaneously [38, 235-238]. Active Appearance Models (AAMs) [38, 238] decompose an image domain into elements (i.e. tetrahedra in 3D) and attach a texture map to each element. Intensity information is analyzed and combined in shape-normalized space (texture space) instead of image space to factor out variability due to differences in geometry. PCA is conducted on both shape and intensity information simultaneously to model variability in
both. Similar to the deformable registration approach, any shape variability not accounted for by the registration algorithm can blur boundaries when images are averaged together [238]. Also, warping images with either deformable image registration or AAMs can stretch or compress edges, which contributes to a change in sharpness.

To illustrate the importance of preserving edge strength, consider a 1D example in which two structures are differentiated by one edge that is slightly misaligned in three images. Figure 78a shows that averaging images (edges located at x={0.25,0.5,0.75}) blurs the edge because the average intensity is computed across different structures. Figure 78b shows that if we first average the location of the edge, shift the intensity profile for each edge to the average edge position, and then average the image intensities, edge shape and strength are maintained. Similarly, warping an image can stretch and compress edges (Figure 79b), but if we warp the geometry and then shift the intensity profiles, edge shape is maintained (Figure 79c). In medical image analysis of cross-sectional or longitudinal differences, it is more often the case image differences are due to variation in the location of boundaries, not the expansion or compression of the geometry. We would like to extend this procedure to 3D; but before doing so, we must have explicit knowledge of edge locations and an intensity representation that shifts along with an edge's location.

![Figure 78: (a) 1D images (solid lines) and average image (dotted line). (b) 1D images after first averaging edge locations (solid lines) and then averaging image intensities (dotted line).](image-url)
Figure 79: (a) original image, (b) warped image, (c) warped parametric image. Notice in (b) that the edges around "NICE 2012" are stretched and edges around "MICCAI" are compressed.

Previous work [6, 7, 12] has focused on improving boundary alignment in registration algorithms so that the scenario in Figure 78b is more likely. However, even if boundaries are perfectly aligned by the registration algorithm, warping an image with a deformation field can stretch or compress edges (Figure 79). This work presents a parametric image representation that explicitly aligns edges before combining intensities so that the scenario in Figure 78b is always the case. The key idea is to decouple shape and intensity information, manipulate them independently, and then synthesize images. We will make the assumption that all edges are step edges (i.e. no texture, shading, or thin lines), and edge sharpness varies as a function of distance to a region's boundary. To demonstrate the utility of such a representation, we construct shape and intensity atlases from MR and CT images of the human body.

### 7.4 Method

#### 7.4.1 Representation

**Geometric Model**

To extend the procedure in Figure 78 and Figure 79 to 3D, a multi-object boundary representation is required to represent edge locations, and a distance field is required to encode intensity information. We use Multi-object Spring Level Sets (MUSCLE) [202, 239] because it provides a memory efficient, sub-voxel precision framework for modeling this type of information (Chapter 6). We now review the MUSCLE representation.
For image domain $\Omega \subset \mathbb{R}^3$ containing labeled object regions $\mathcal{L} = \{1, \cdots, L\}$, the MUSCLE representation consists of a label mask $\chi: \Omega \rightarrow \mathcal{L}$, distance field $\psi: \Omega \rightarrow \mathcal{R}$, and constellation of triangles with vertices $q_n \in \mathbb{R}^3$. Each object has a corresponding constellation of triangles that represent its boundary. To identify objects in the aggregate constellation, a label $l_n \in \mathcal{L}$ is associated with each triangle vertex $q_n$. Constellations are coupled with signed distance fields $\phi_l: \Omega \rightarrow \mathcal{R}$ that provide a redundant, implicit representation of each object $l$. Since an image may have hundreds of different regions, level sets are compressed into a "distance field + label mask" representation. The distance field image measures the unsigned distance to the closest object boundary at each voxel (i.e. $\psi(x) = \min_l |\phi_l(x)|$), and the label mask image indicates to which object a voxel belongs. The smallest label is used in the event that a voxel is inside more than one object (i.e. $\chi(x) = \min_{l\mid |\phi_l(x)| < 0} l$). This geometric representation is sufficient to reconstruct the boundaries of each object with sub-voxel precision. It is formally defined as the set of data structures $\mathcal{M}_g = \{\psi, \chi, \{q_1, \cdots, q_N\}, \{l_1, \cdots, l_N\}\}$.

In the MUSCLE framework, deformations are applied to a constellation of triangles; after which, the distance field and label mask are evolved to track the moving mesh with Multi-Object Geodesic Active Contours (MOGAC) [240] (Chapter 4). MUSCLE is an efficient way to parametrically represent and evolve multiple objects that guarantees no overlaps, self-intersections, or air-gaps between adjacent structures. It is also trivial to determine whether a voxel is inside a particular object with $\chi(x)$ and determine the distance from each voxel to the nearest boundary with $\psi(x)$. These two properties will be used to attach intensity information to $\mathcal{M}_g$ that accounts for mixtures (partial volumes) of tissue classes that occur near the boundary of object regions.

**Intensity Model**

To model edge intensity profiles analogous to the step edges in Figure 78b, we compute the average intensity for voxels within distance ranges $Q = \{(0, 1), [1, 2), [3, 4), [4, \infty)\}$ to the closest object boundary and store them in the function $f(l, d)$, where $l$ is the object label and $d$ is the distance to the closest object boundary (see Figure 80). This model accounts for some of the inhomogeneity in image intensities that occur near the boundary of object regions as a function of distance. However, the model assumes only two objects are involved in the mixture.
This assumption holds for the White Matter (WM) / Gray Matter (GM) and GM/Dura interfaces in the brain, but does not hold at the WM / (sub-cortical GM, cerebellum, ventricle) interfaces.

Figure 80: Intensity as function of distance \( f(l,d) \) to an object boundary for object \( l \).

We would like to attach more intensity information to \( M_g \) without introducing more geometric data structures. One useful extension is to model intensity mixtures for pairs of objects along object boundaries. To do so, voxels \( y \in \Lambda \) at object boundaries \( \Lambda = \{ y | \exists x \in \mathcal{N}(y) \text{ s.t. } \chi(x) \neq \chi(y) \} \) are located where \( \mathcal{N}(x) \) is the 18-connected neighborhood around \( x \). For each location \( x \), we identify location \( x^* \in \mathcal{N}(x) \), which is the point on the closest neighboring object. The average intensity in image \( I: \Omega \to \mathcal{R} \) for each label pair \((a, b)\) is \( g(a, b) = \frac{1}{|H|} \sum_{x \in H} I(x) \) where \( H = \{ x | a = \chi(x) \text{ and } b = \chi(x^*) \} \) (see Figure 81). Note that this definition is not symmetric near the boundary between more than two objects.

Figure 81: Depiction of 3 objects. The 4 voxels \( \{x_1, x_2, x_3, x_4\} \) are associated with edges pairs \( x_1 \rightarrow (l,n), x_2 \rightarrow (m,l), x_3 \rightarrow (n,m), x_4 \rightarrow (n,l) \). The intensities at these voxel locations are used in the average for their edge pairs.
Given a label image, distance field, and model $\mathcal{M}_p = \{f, g\}$, the intensity of a pixel is,

$$I(x) = \begin{cases} g(\chi(x), \chi(x^*)) & x \in \Lambda \\ f(\chi(x), \psi(x)) & x \notin \Lambda \end{cases}.$$  

(7.1)

This model assumes objects enclose homogeneous regions of intensity with the possibility that intensities near the boundary are a mixture from at most two objects. The model is capable of recovering all the intensity information in the image as $L \rightarrow |\Omega|$, but we would like to keep $L$ small so that the representation is compact.

**Linear Transforms**

Before parametric images (p-images) can be combined, we must ascribe a linear structure to the space of image models $\mathcal{M}_p = (\mathcal{M}_g, \mathcal{M}_f)$. Given two parametric images $A, B \in \mathcal{M}_p$ and scalars $\alpha, \beta \in \mathcal{R}$, the parametric image $C = \alpha A + \beta B$ is computed by independently transforming $\mathcal{M}_g$ and $\mathcal{M}_f$. Since point correspondences are maintained in the MUSCLE framework when resampling is disabled, mesh vertices for $C$ are computed by $q^C_n = \alpha q^A_n + \beta q^B_n$ and vertex labels computed by $l^C_n = l^A_n$. After determining mesh vertices and vertex labels, the distance field and label mask are computed by rasterizing each triangle constellation to a level set $\varphi_I$ and then combining level sets into $\psi_C(x) = \min_l |\varphi_I(x)|$ and $\chi_C(x) = \min_l(|\varphi_I(x)| < 0) \cdot l$. The computational bottleneck for p-images is in rasterizing triangle meshes to level sets. Rasterization is implemented on the CPU in this work, but there are GPU accelerated methods for performing this task [191]. The final geometric model is $\mathcal{M}^C_g = \{\psi_C, \chi_C, \{q^C_1, \cdots, q^C_N\}, \{l^C_1, \cdots, l^C_N\}\}$. It is straightforward to combine intensity information since $g: \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{R}$ and $f: \mathcal{L} \times \mathcal{R} \rightarrow \mathcal{R}$. The final intensity model is $\mathcal{M}^C_i = \{\alpha f_A + \beta f_B, \alpha g_A + \beta g_B\}$. Equipped with a linear structure on the space of parametric image models, we can form linear combinations of p-images; and because intensity varies as a function of distance to a region boundary, it is simple to shift intensity profiles along with edge locations as in Figure 78 and Figure 79. $\Omega_p$ will be used in subsequent sections to analyze shape and intensity information with PCA.
7.4.2 Parametric Image Construction

There are several ways to construct parametric images. To construct parametric images (p-images) of the brain, we process MR images with Topology-preserving, Anatomy Driven Segmentation (TOADS) to produce a hard classification image consisting of 10 tissue classes [241]. The hard classification is then converted to a "distance field + label mask" representation, and MOGAC is used to smooth object boundaries with mean-curvature flow [66]. It is straightforward to compute $\mathcal{M}_I$ from the MR image once the distance field and label mask are aligned with mesh boundaries (see the Intensity Model sub-section in 2.1). As shown in Figure 82, the final model captures most of the shape and intensity information present in the MR image. Another approach is to use an existing geometric phantom (e.g. mesh, level set, or region mask) with a corresponding synthesized medical image (like the XCAT [230]) and re-express it as a p-image (Figure 86a-b).

7.4.3 Parametric Image Warping

As a prerequisite for atlas construction, the following describes how to warp parametric images using displacement fields produced by existing registration algorithms. Although it is likely beneficial to incorporate p-images into the registration process, we do not want to restrict p-image usage to any particular registration method. Instead, affine transformations and displacement fields are applied to p-images with techniques developed in the MUSCLE framework [239]. Given an affine transformation $A$ produced by a global registration algorithm, such as FLIRT [5], the transformation is first applied to the constellation of triangles, label mask, and distance field. The image re-sampling process required to transform the label mask and distance field introduces distortion in the level set functions $\varphi_l$ implied by the "label mask + distance field" representation. This distortion is corrected by evolving level sets to minimize the distance from their iso-surfaces to the constellation of triangles. To do so, the image $\omega: \Omega \rightarrow \mathcal{R}$ is first computed, which measures the clamped unsigned distance to the constellation of triangles:

$$\omega(x) = \min\{2d_{\text{max}}, d_1(x) \ldots d_N(x)\}$$ (7.2)
where \( d_k(x) \) is the distance from location \( x \) to triangle \( k \), and \( d_{\text{max}} = 0.5 \) is the clamped distance. Level sets \( \varphi_i \) are then evolved to minimize the following objective function:

\[
E = \sum_i \int \left( \frac{1}{2} (\omega(x))^2 + \lambda |\nabla \varphi_i(x)| \right) \delta(\varphi_i(x)) dx
\]  

(7.3)

where \( \lambda \) is a regularization weight that controls the model’s smoothness. Instead of evolving \( \varphi_i \)'s directly, the label mask and distance field are evolved with MOGAC [240]. Global registration is followed by deformable registration, which produces a displacement field \( \vec{v}: \Omega \rightarrow \mathcal{R}^3 \) describing where locations in the source image map to in the target image. The geometry of a p-image is warped with the displacement field by incrementally advecting mesh vertices \( q_n \) from source to target with \( q_n(t) = q_n(0) + t \vec{v}(q_n(0)) \) where \( t \in [0,1] \). After each displacement step \( k \) s.t. \( t = k \Delta t \) for \( k = 0, 1, ..., [1/\Delta t] \), the label mask and distance field are evolved to track the moving mesh. The step size is chosen to be \( \Delta t \leq d_{\text{max}}/\max_n \| \vec{v}(q_n) \| \) so that the level sets stay within the capture range of the clamped distance field \( \omega \).

One alternative to level set tracking is to simply apply the displacement field to the constellation of triangles, rasterize the mesh to a collection of level sets, and compress the level sets into a label mask and distance field. The computational complexity of rasterizing \( L \) level sets to an \( M \times M \times M \) image is \( O(LM^3) \), whereas the step complexity for evolving \( L \) level sets with MOGAC is \( O(M^2) \). Level set evolution is usually faster because the number of steps \( K = [1/\Delta t] \ll M \). However, rasterization is necessary when synthesizing a p-image from a Point Distribution Model (PDM) [37], which we will describe in the subsequent section.

### 7.4.4 Atlas Construction

Given a template image and collection of subject images, the atlas construction pipeline begins by constructing a p-image for the template with the methods presented in section 2.2. The template is affine registered to each subject [5] and then deformably registered with the Mutual Information based method from Rohde et al. [11]. The affine transform and displacement field are applied to the template p-image with the method described in section 2.3. A Point Distribution Model [37] is then constructed for the warped meshes. The mean triangle constellation is rasterized to a label mask and distance field to construct an average geometry.
model $M_g$. Since the warped p-images have the same intensity model as the template, the average intensity model $M_I$ is the intensity model for the template. An intensity model for each subject is constructed using the intensities for each subject after the geometry model for the template has been warped into the space of each subject image. The full procedure is as follows:

1. Construct a p-image for the template using an existing segmentation.
2. Affine register each subject's image to the template.
3. Deformably register the affine registered image from step 2 to the template.
4. Apply the affine transformation and displacement field to the template's p-image.
5. After warping the template geometry model $M_g$ into the space of individual subjects, compute the intensity model $M_I$ for each subject.
6. Average subject p-images to construct an average intensity and geometry model.
7. Rasterize the average geometry model to a label mask a distance field.
8. Construct a Point Distribution Model using the geometry model for each subject.

7.5 Results

7.5.1 Human Brain Atlas

A human brain atlas was constructed from 19 normal subjects chosen at random from the OASIS MRI database [168]. The BrainWeb phantom [242] with no-noise or inhomogeneity was used for the template image. Note that the template is a simulation of a T1 weighted SPGR sequence and the subjects were imaged with a T1 weighted MPRAGE sequence. Therefore, their intensities follow slightly different statistical distributions. An advantage of p-images is that either intensity model (SPGR or MPRAGE) can be attached to the geometric model. Table 1 summarizes all registration results. MR image and p-image registration results look slightly different (Figure 83) because p-images and MR images use different displacement fields. P-image deformation is a Lagrangian technique that uses the forward source-to-target displacement field whereas MR image deformation is a semi-Lagrangian technique that uses the target-to-source displacement field. Displacement fields produced by target-to-source and source-to-target registration are usually not inverses of each other. Looking at MR registration to the template and p-image registration to the subject, both of which use the source-to-target displacement field,
Normalized Mutual Information (NMI) is the same in both cases (Table 1). Looking at MR registration to the subject and p-image registration to the template, both of which use the target-to-source displacement field, the NMI is slightly higher for p-images.

Figure 84 shows a mean intensity atlas using either MR Images or P-Images. Although both atlases are similar (Correlation Coefficient of 0.95), the p-image atlas is noticeably sharper. To quantitatively assess image quality, we ran the CRUISE cortical reconstruction pipeline [226] on the BrainWeb phantom, MR Image atlas, and p-image atlas. The cortical reconstruction produced by segmenting the p-image atlas is closer (mean surface distance of 0.37±0.35 mm) to the BrainWeb WM/GM surface than the reconstruction from the MR image atlas (surface distance of 0.88±0.71 mm). To synthesize images that deviate from the mean p-image, mode weights were specified to generate a mesh. After which, the mesh was rasterized to a label mask and distance field. Figure 85 shows both the geometry and synthesized images for the first PCA mode with the mean intensity model. Synthesized images from the p-image atlas are consistently sharp and represent both shape and intensity information. These results suggest p-images could easily substitute PDMs in existing registration and segmentation algorithms.

Table 17. Registration results for different image representations and registration targets. The table reports Normalized Mutual Information (NMI) between the registered image and target. Regions outside the brain were excluded in these measurements.

<table>
<thead>
<tr>
<th>Representation</th>
<th>Template (SPGR)</th>
<th>Subject (MPRAGE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR Image</td>
<td>1.12±0.01</td>
<td>1.08±0.01</td>
</tr>
<tr>
<td>P-Image (SPGR)</td>
<td>1.10±0.02</td>
<td>1.12±0.01</td>
</tr>
<tr>
<td>P-Image (MPRAGE)</td>
<td>1.10±0.02</td>
<td>1.12±0.01</td>
</tr>
</tbody>
</table>
Figure 82: (a) BrainWeb template image, (b) p-image with just distance intensity model $\mathcal{M}_f = \{f\}$, and (c) p-image with distance and edge intensity model $\mathcal{M}_f = \{f, g\}$. The Correlation Coefficients between (a)/(b) and (a)/(c) are 0.91 and 0.93 respectively.

Figure 83: (a) target image, (b) registered MR Image, and (c) registered p-image.

Figure 84: (a) BrainWeb WM/GM surface, (b) mean of registered MR images, (c) WM/GM surface reconstructed from (b), (d) mean intensity p-image, (e) WM/GM surface reconstructed from (d). Mean error in (c) is $0.88 \pm 0.71$ mm. Mean error in (e) is $0.37 \pm 0.35$ mm. Correlation Coefficient between (b) and (d) is 0.95.
7.5.2 Human Body Atlas

A human body atlas was constructed from 20 whole-body CT images (12 adult males, 8 adult females) using the XCAT phantom [230] as a template. The XCAT is a whole-body NURB surface representation from which a CT image is simulated (Figure 86a). We used an instance of the phantom for the 50 percentile male consisting of 50 structures. NURB surfaces for each structure were converted to triangle meshes and then rasterized to a label mask and distance field. After which, an image intensity model was constructed to form a p-image representation for the XCAT (Figure 86b). Each subject was registered to the XCAT's CT image using the same procedure and methods for the human brain. Then, the p-image template was warped into the space of each subject. PCA analysis was conducted on the p-images. Fig 6. compares the mean intensity image achieved by warping CT images and warping p-images. One can surmise from Figure 86c that the registration algorithm did not align bones well because these regions are blurry and bear little resemblance to bone geometry. However, averaging p-image intensity models (Figure 86d) preserves bone geometry in these regions. Figure 87 shows the mean and plus / minus the first (largest) shape mode. The MUSCLE representation captures variation in all 50 structures simultaneously and because the structures do not overlap or have air-gaps, it is straightforward to simulate a CT image based on $\mathcal{M}_p$ (eq. 1).
Figure 86: (a) XCAT CT image, (b) p-image constructed from XCAT CT image, (c) mean CT image after registration, (d) mean intensity p-image after registration. The correlation coefficient between (a)/(b) is 0.96 and between (c)/(d) is 0.97. Image size: $256 \times 256 \times 512$. We thank Dr. Ben Tsui and Dr. George Fung for providing the XCAT and Dr. John Carrino for providing data.
Figure 87: P-image renderings showing the geometry (top) and synthesized CT images (bottom) for (a/d) mean minus one std. dev., (b/e) mean, (c/f) mean plus one std. dev. Computation time for a $256 \times 256 \times 256$ image and 6M triangles was 5.5 min on a PC with dual Intel X5450s.

7.6 Conclusion

This work has presented a new data structure and linear operator for representing and manipulating images that preserves edge strength. To demonstrate, we applied p-images to deformable registration and construction of an atlas for the human brain and human body. Parametric images assume all edges are step edges and structures do not contain texture, shading, or thin lines. These assumptions do not hold in general, but for MR images of the human brain (after inhomogeneity correction) and CT images of the human body, p-images are highly correlated with the real images (Correlation coefficient of 0.93 for the human brain and 0.96 for the human body.

One claim we make is that p-images are consistently sharp when synthesizing images from an atlas. This appears to be the case, at least subjectively (Figure 85). One future direction would be to quantify sharpness under some metric. However this is non-trivial because there are more
than 13 definitions of sharpness [243]. Another option would be to look at the preservation of edge strength under non-rigid image transformations with the following metric $P(I)$:

$$P(I)^2 = \frac{1}{|\Omega|} \sum_{\mathbf{x}} (T(\nabla I)(\mathbf{x}) - \nabla T(I)(\mathbf{x}))^2$$

(7.4)

where $I: \Omega \to \mathbb{R}$ is an image, $T(\cdot)(\mathbf{x})$ is an image transformation, and $\nabla I(\mathbf{x})$ is the gradient of the image. This metric assumes images are smooth and edges differentiate between at most two objects (i.e. no junctions that involve more than 2 objects). We found it difficult to use this metric in the human brain and human body examples because they violate both these assumptions. The sparse-field approximation to the distance field is not smooth, and there are many locations in both examples where more than two objects share boundaries.

Another future direction would be to model statistical variation in intensity that occurs within a structure. For instance, Active Appearance Models and other statistical methods, similar to those described in the background (section 7.3), could be applied to p-images to model variation in shape and intensity information simultaneously with PCA. Finally, it would be interesting to incorporate a p-image atlas into registration and segmentation pipelines to see if there is any improvement in performance. An interesting feature of p-images is that they can model articulated structures because the MUSCLE representation allows surfaces to slide on one another. For this reason, p-images may be advantageous when segmenting articulated structures like bone joint regions in the human body.

**Contributions:**

- We have presented a new parametric image representation in which shape and intensity information can be manipulated independently.
- We have described how to warp parametric images in a way that preserves edge strength.
- We have described a linear operator that can be used to combine parametric images.
- We have used parametric images to construct atlases of the human brain and human body that capture variations in shape and appearance.
Chapter 8

8 Conclusion

8.1 Summary

This work has presented new geometric data structures to provide interoperability between image analysis methods. MOGAC (Chapter 4), SpringLS (Chapter 5), and MUSCLE (Chapter 6) merge together existing data structures so that methods designed for one type can be applied to a method intended for a different type of data structure. The importance of interoperability is highlighted by several imaging applications in which registration and active contour segmentation are used in combination to produce better segmentation results than could be obtained with either method independently.

MUSCLE has sub-voxel precision, is parametric, re-meshes, tracks point correspondences, and guarantees no self-intersections, air-gaps, or overlaps between adjacent structures. No shape information is lost in the transformation from triangle mesh, level set, or label mask into MUSCLE because the representation is simultaneously all three. A potential drawback of the representation is that because it has to maintain three data structures, it has the computational and memory overhead of all three. To overcome this additional overhead, MUSCLE is implemented in OpenCL to leverage parallelism now abundant on modern CPUs and GPUs. The speed at which a MUSCLE can be deformed is limited by level set evolution. To address this limitation, we have presented a new multi-object level set method (MOGAC) that is faster and uses less memory than existing algorithms.

MUSCLE is well positioned to address challenges of emerging applications in image analysis. In particular, simultaneous segmentation of hundreds to thousands of objects in 3D data. We have demonstrated how to simultaneously segment 74 different structures in the human brain and 50 structures in the human body using MUSCLE. The utility of segmenting almost all structures visible in medical images is demonstrated by a new image warping and atlas construction method (Chapter 7).
8.2 Future Work

- We would like to understand better the interplay between relaxation and re-sampling in Spring Level Sets. Although empirical evidence has been presented that shows the re-sampling process converges to a stable springls constellation, we have no proof of convergence.

- The SpringLS implementation assumes the springls constellation needs to be re-sampled everywhere all the time. This is usually not the case because many times, only a portion of the mesh is deforming. An obvious speed-up would be to implement local re-sampling so that only regions undergoing large deformations are relaxed and/or re-sampled.

- SpringLS can only maintain injective point mappings when re-sampling is enabled. Point Distribution Models require a bijective mapping, so new methods are required for either constructing Point Distribution Models based on injective mappings or transforming injective mappings into bijective ones.

- Coupling SpringLS with finite element methods via embedded mesh techniques would unify the most popular data structures used in image analysis and physical simulation. The implications of a unified model would be interesting to explore. In particular, using imaging segmentations to validate physical simulations, and using physical simulations to improve the robustness of image segmentation systems.

- Whole body segmentation was introduced at the end of Chapter 6 as an open problem in the imaging field that benefits from MUSCLE. We would like to improve and extend that system in future work and possibly incorporate a parametric image atlas.

- Current systems for manually segmenting images operate in 2.5 dimensions (i.e. the user paints on 2D image slices displayed in Triplanar view). The SpringLS data structure gives level sets "handles" with which to pull and push on the surface to cause deformations. This has the potential to allow users to sculpt surfaces as in 3D tools like Sculptris. An interesting application of SpringLS would be to build a 3D semi-automated segmentation tool that combines 3D sculpting with image segmentation.

---

Appendix

Spring Level Set Relaxation as a Physical System

To find the gradient of potential energy

\[ VE(S_n) = \sum_m \sum_k \frac{1}{2} \kappa_{\text{ext}} \| q_{n,m} - q_{n,m,k} \|^2 + \sum_m \frac{1}{2} \kappa_{\text{int}} (l_{n,m} - 2r)^2 \]  \hspace{1cm} (A.1)

with respect to \( l_{n,m} \) and \( \vec{b}_n \), we rewrite the potential energy in a local coordinate frame (eq. 2)

where \( \vec{q}_{n,m,k} = M(-\vec{b}_n)(q_{n,m,k} - p_n) \).

\[ VE(S_n) = \sum_m \sum_k \frac{1}{2} \kappa_{\text{ext}} \| l_{n,m} M(\vec{b}_n) \vec{s}_{n,m} - \vec{q}_{n,m,k} \|^2 + \sum_m \frac{1}{2} \kappa_{\text{int}} (l_{n,m} - 2r)^2 \]  \hspace{1cm} (A.2)

Applying the law of cosines and rearranging terms we obtain

\[ VE(S_n) = \sum_m \sum_k \frac{1}{2} \kappa_{\text{ext}} \left\{ \| \vec{q}_{n,m,k} \|^2 + l_{n,m}^2 - 2l_{n,m} (M(\vec{b}_n) \vec{s}_{n,m}) \cdot \vec{q}_{n,m,k} \right\} + \sum_m \frac{1}{2} \kappa_{\text{int}} (l_{n,m} - 2r)^2 . \]  \hspace{1cm} (A.3)

Differentiating with respect to \( l_{n,m} \), we obtain

\[ \frac{\partial VE}{\partial l_{n,m}} = -\kappa_{\text{ext}} \sum_k \left( (M(\vec{b}_n) \vec{s}_{n,m}) \cdot \vec{q}_{n,m,k} - l_{n,m} \right) + \kappa_{\text{int}} (l_{n,m} - 2r), \]  \hspace{1cm} (A.4)

which is equivalent to

\[ \frac{\partial VE}{\partial l_{n,m}} = (M(\vec{b}_n) \vec{s}_{n,m}) \cdot (f_{n,m}^\text{int} + f_{n,m}^\text{ext}). \]  \hspace{1cm} (A.5)

To find the partial derivative w.r.t. \( \vec{b}_n \), we linearly approximate \( M(\vec{b}_n) \):

\[ (M(\vec{b}_n) \vec{s}_{n,m}) \cdot \vec{q}_{n,m,k} = \vec{q}_{n,m,k} \cdot \vec{s}_{n,m} + \vec{b}_n \cdot \vec{s}_{n,m} \times \vec{q}_{n,m,k} + O(\| \vec{b}_n \|^2) \]  \hspace{1cm} (A.6)

Substituting eq. 6 into eq. 2 and solving for the partial derivative, we obtain

\[ \frac{\partial VE}{\partial \vec{b}_n} = -\kappa_{\text{ext}} \sum_m l_{n,m} \vec{s}_{n,m} \times \vec{q}_{n,m,k}, \]  \hspace{1cm} (A.7)

which is equivalent to

\[ \frac{\partial VE}{\partial \vec{b}_n} = \sum_m l_{n,m} \vec{s}_{n,m} \times f_{n,m}^\text{ext}. \]  \hspace{1cm} (A.8)
Spring Level Set Relaxation as a Non-Physical System

The non-physical potential energy equation is

$$VE^*(\mathbf{s}_n) = \sum_m \frac{1}{2} \| \mathbf{g}_{n,m} \|^2 + \sum_m \frac{1}{2} \kappa_{\text{int}} (l_{n,m} - 2r)^2 \tag{A.9}$$

The gradient of $VE^*(\mathbf{s}_n)$ is given by

$$\frac{\partial VE^*}{\partial l_{n,m}} = \frac{\partial q_{n,m}}{\partial l_{n,m}} \hat{f}_{\text{int}} + \langle \mathbf{g}_{n,m}, \mathbf{J}_{n,m} \frac{\partial q_{n,m}}{\partial l_{n,m}} \rangle,$$ \hspace{1cm} \tag{A.10}$$

and

$$\frac{\partial VE^*}{\partial \mathbf{b}_n} = \sum_m \langle \mathbf{g}_{n,m}, \mathbf{J}_{n,m} \frac{\partial q_{n,m}}{\partial \mathbf{b}_n} \rangle,$$ \hspace{1cm} \tag{A.11}$$

where $\mathbf{J}_{n,m}$ is the $3 \times 3$ Jacobian of $\mathbf{g}_{n,m}$ with respect to $q_{n,m}$. Rearranging terms, we obtain

$$\frac{\partial VE^*}{\partial l_{n,m}} = (\mathbf{M}(\mathbf{b}_n) \mathbf{s}_{n,m}) \cdot (\hat{f}_{\text{int}} + \mathbf{J}_{n,m}^T \mathbf{g}_{n,m}) \tag{A.12}$$

and

$$\frac{\partial VE^*}{\partial \mathbf{b}_n} = \sum_m l_{n,m} \mathbf{s}_{n,m} \times (\mathbf{J}_{n,m} \mathbf{g}_{n,m}) \tag{A.13}$$
References


M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *SGP '05*, 2005, pp. 73-.

A. Hornung and L. Kobbelt, "Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information," in *SGP '06*, 2006, pp. 41-50.


Blake C. Lucas earned Bachelor of Science degrees in Computer Science, Electrical Engineering, and Computer Engineering from North Carolina State University in 2006. He then earned Master of Science degrees in Electrical Engineering and Computer Science from Johns Hopkins University in 2007 and 2010 respectively before entering Ph.D. candidacy. He is a member of the Laboratory for Computational Sensing and Robotics (LCSR) at Johns Hopkins University and the Biomedical and Image-Guided Surgical Systems (BIGSS) at the JHU Applied Physics Laboratory. Blake is interested in real-world applications of deformable models that involve multiple objects with many geometric constraints and priors. His research focuses on new types of deformable models and their application to problems in medical image analysis, computer vision, and computer graphics.