Summer 2017

Finite Element Modeling Driven by Health Care and Aerospace Applications

Fotios Drakopoulos
Old Dominion University, fdrakopo@gmail.com

FINITE ELEMENT MODELING DRIVEN BY HEALTH CARE AND AEROSPACE APPLICATIONS

by

Fotios Drakopoulos
B.A. August 2004, University of Patras, Greece
M.A. May 2008, University of Patras, Greece

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2017

Approved by:

Nikos P. Chrisochoides (Director)

Andrey Chernikov (Member)

Yaohang Li (Member)

Charles I. Sukenik (Member)

Mike Park (Member)

Sarah F. Frisken (Member)

ABSTRACT

FINITE ELEMENT MODELING DRIVEN BY HEALTH CARE AND AEROSPACE APPLICATIONS

Fotios Drakopoulos
Old Dominion University, 2017
Director: Dr. Nikos P. Chrisochoides

This thesis concerns the development, analysis, and computer implementation of mesh generation algorithms encountered in finite element modeling in health care and aerospace. The finite element method can reduce a continuous system to a discrete idealization that can be solved in the same manner as a discrete system, provided the continuum is discretized into a finite number of simple geometric shapes (e.g., triangles in two dimensions or tetrahedrons in three dimensions).

In health care, namely anatomic modeling, a discretization of the biological object is essential to compute tissue deformation for physics-based simulations. This thesis proposes an efficient procedure to convert 3-dimensional imaging data into adaptive lattice-based discretizations of well-shaped tetrahedra or mixed elements (i.e., tetrahedra, pentahedra and hexahedra). This method operates directly on segmented images, thus skipping a surface reconstruction that is required by traditional Computer-Aided Design (CAD)-based meshing techniques and is convoluted, especially in complex anatomic geometries. Our approach utilizes proper mesh gradation and tissue-specific multi-resolution, without sacrificing the fidelity and while maintaining a smooth surface to reflect a certain degree of visual reality.

Image-to-mesh conversion can facilitate accurate computational modeling for biomechanical registration of Magnetic Resonance Imaging (MRI) in image-guided neurosurgery. Neuronavigation with deformable registration of preoperative MRI to intraoperative MRI allows the surgeon to view the location of surgical tools relative to the preoperative anatomical (MRI) or functional data (DT-MRI, fMRI), thereby avoiding damage to eloquent areas during tumor resection. This thesis presents a deformable registration framework that utilizes multi-tissue mesh adaptation to map preoperative MRI to intraoperative MRI of patients who have undergone a brain tumor resection. Our enhancements with mesh adaptation improve the accuracy of the registration by more than 5 times compared to rigid and traditional physics-based non-rigid registration, and by more than 4 times compared

to publicly available B-Spline interpolation methods. The adaptive framework is parallelized for shared memory multiprocessor architectures. Performance analysis shows that this method could be applied, on average, in less than two minutes, achieving desirable speed for use in a clinical setting.

The last part of this thesis focuses on finite element modeling of CAD data. This is an integral part of the design and optimization of components and assemblies in industry. We propose a new parallel mesh generator for efficient tetrahedralization of piecewise linear complex domains in aerospace. CAD-based meshing algorithms typically improve the shape of the elements in a post-processing step due to high complexity and cost of the operations involved. On the contrary, our method optimizes the shape of the elements throughout the generation process to obtain a maximum quality and utilizes high performance computing to reduce the overheads and improve end-user productivity. The proposed mesh generation technique is a combination of Advancing Front type point placement, direct point insertion, and parallel multi-threaded connectivity optimization schemes. The mesh optimization is based on a speculative (optimistic) approach that has been proven to perform well on hardware-shared memory. The experimental evaluation indicates that the high quality and performance attributes of this method see substantial improvement over existing state-of-the-art unstructured grid technology currently incorporated in several commercial systems. The proposed mesh generator will be part of an Extreme-Scale Anisotropic Mesh Generation Environment to meet industries expectations and NASA's CFD vision for 2030.

ACKNOWLEDGEMENTS

I would like to thank my parents for always being there for me. I would also like to express the deepest appreciation to my advisor Professor Nikos Chrisochoides for his guidance and support. None of this work would be possible without his insight and encouragement. Special thanks to the dissertation committee members Dr. Andrey Chernikov, Dr. Yaohang Li, Dr. Charles I. Sukenik, Dr. Mike Park, and Dr. Sarah F. Frisken for their constructive suggestions and discussions. I would like to thank all my research collaborators (Dr. Chengjun Yao, Dr. Ron Kikinis, and Dr. Kathryn Holloway) and previous colleagues (Dr. Yixun Liu, Dr. Panagiotis Foteinos, Dr. Andrey Fedorov, Dr. Andriy Kot, and Mike Weissberger) for fruitful discussions, as well as the Computer Science Department of Old Dominion University for providing the tools necessary for my research. Many thanks also to Caitlin Reed for proofreading this dissertation. This work is funded in part by NASA grant no. NNX15AU39A, DoDs PETTT Project PP-CFD-KY07-007, NSF grants CCF-1139864 and CCF-1439079, John Simon Guggenheim Foundation, ODU Modeling and Simulation fellowship, and Richard T. Cheng Endowment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Finite Element Method (FEM) [5, 192, 46] has found wide appeal in solving complex physical problems in science and engineering in the last few decades. A physical problem typically involves a continuous domain subjected to external loads. The idealization of the physical problem to a mathematical model requires certain assumptions that together lead to partial differential equations (PDEs) governing the mathematical model. Explicit closed-form solutions for PDEs are rarely available. Nevertheless, if the continuous domain is discretized into a finite number of simple geometric shapes (e.g., triangles or quadrilaterals in 2-dimensions and tetrahedra, pentahedra, or hexahedra in 3-dimensions) then together with the use of FEM [5] and the Principle of Virtual Work [15] the continuous-system mathematical model can be reduced into a discrete idealization, which can be solved as a discrete-system using numerical procedures. The collection of those simple geometric shapes is called *mesh* or *grid*. A computer program that discretizes a domain into simple geometric shapes is called a *mesh generator* or *grid generator*.

This thesis concerns the development, analysis, and computer implementation of mesh generation algorithms encountered in finite element modeling. In general, a mesh generation method should satisfy two main requirements. First, it should discretize the domain with well-shaped elements. The shape of the elements is critical for the accuracy and the convergence of a finite element solution. For example, elements with large dihedral angles tend to increase the discretization error in the solution [8]. On the other hand, elements with small dihedral angles are bad for matrix conditioning but not for interpolation or discretization [74, 173]. Unstructured grids contain elements whose shape vary significantly. The problem is more severe in 3-dimensions because tetrahedra can be distorted in more ways compared to triangles in 2-dimensions (Figure 1).

Second, a mesh generation algorithm should satisfy the time constraints imposed by the analysis. Mesh generation remains one of the most time consuming aspects of numerical simulations of complex configurations [155, 156] (Figure 2). The cost can be higher if validation and optimization of products and manufacturing require re-meshing. For example, if solution accuracy criteria are not met, the numerical solution has to be repeated with a refined mesh until a sufficient accuracy is reached. Besides, applications

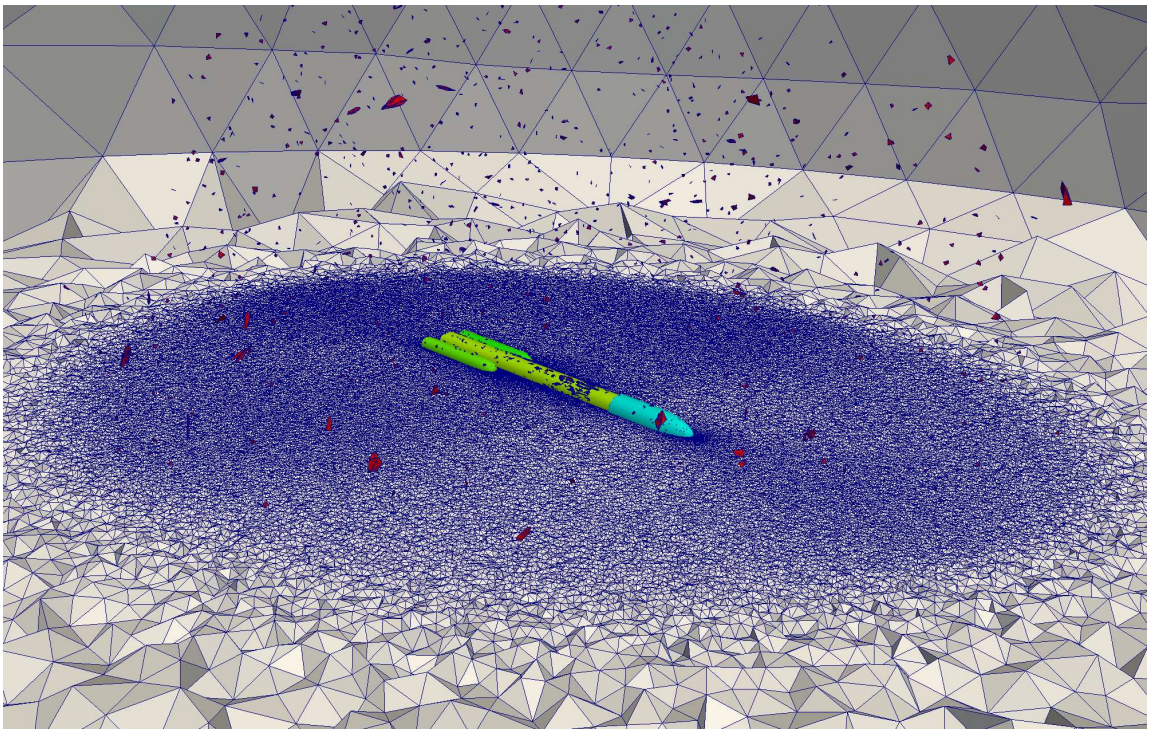


Figure 1: Cut of tetrahedral mesh of the flow field of two-booster launch vehicle with 100 M elements. Red signifies the badly-shaped elements with a dihedral angle smaller than 2° or larger than 178°. As the vertices of a badly-shaped element approach coplanarity, the finite element mapping [15] becomes unreliable, affecting the accuracy of the numerical solution. This thesis presents a new mesh generator that incorporates parallel connectivity optimization and other mechanisms to eliminate badly-shaped elements.

such as biomechanical deformable registration for image-guided neurosurgery [58, 57] and interactive surgical simulations [59] impose hard time constraints in the mesh generation process. Application of High Performance Computing (HPC) in the simulation workflow will surely shorten time to obtain results. In some cases, this has an enabling effect as certain kinds of calculations become achievable within timescales typical in commercial engineering projects. Additionally, HPC opens a path towards automated optimization, where alternative design variants can be tested with limited need for human intervention. This dissertation focuses both on quality and high-performance mesh generation.

The developed algorithms are driven by applications in health care and aerospace. In health care, namely anatomic modeling, image-to-mesh conversion algorithms are essential in modeling tissue deformation using FEM. This has significant implications in many areas, such as image-guided therapy [185, 4, 58], interactive surgery simulation

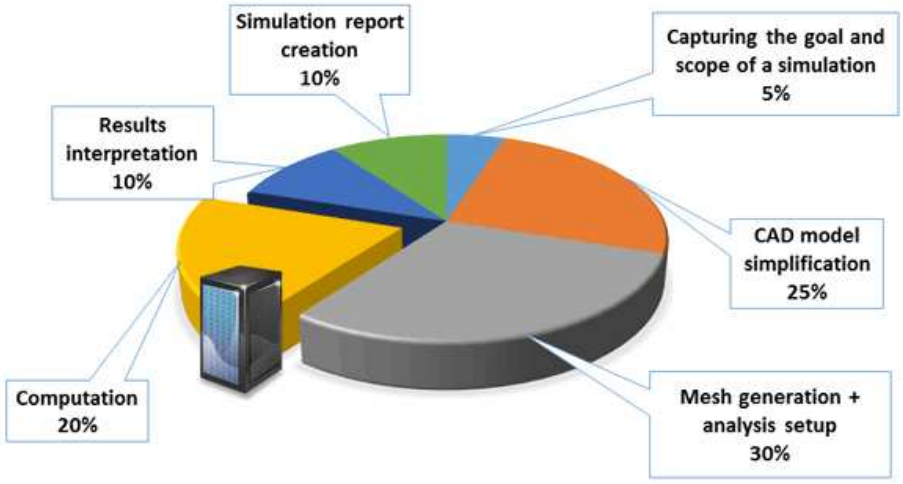


Figure 2: Time required to perform the stages of a finite element simulation (courtesy of forkedbranch SOFTWARE RESEARCHED). This thesis employs high performance computing to reduce the cost of mesh generation without sacrificing well-shaped elements critical for the accuracy of the numerical solution.

for training young clinicians [59], and endovascular flow diversion for aneurysm geometries [53, 54, 101].

This thesis augments a previous meshing technique [65, 119] to accurately convert 3-dimensional imaging data into adaptive lattice-based discretizations of well-shaped elements. The proposed method operates directly on isosurface-based data (i.e., segmented multi-labeled images), therefore skipping a surface reconstruction that is required by traditional Computer-Aided Design (CAD)-based meshing techniques. Surface reconstruction can be time consuming, lacking robustness, and virtually intractable for complex anatomic geometries, such as stented arterial segments [53] or brain Arteriovenous Malformations (AVM) [59]. CAD-based meshing methods fail to exploit the fact that data surfaces are not explicitly defined for 3-dimensional imaging data but rather implicitly as the surface boundaries of a segmented volume of interest.

The present image-to-mesh conversion method utilizes proper mesh gradation and tissue-specific multi-resolution, without sacrificing the fidelity and while maintaining a smooth surface to reflect a certain degree of visual reality. Specifically, this method offers several advantages compared to previous implementation [119]: (i) it improves the accuracy of the geometric and topologic representation; (ii) it provides material-dependent mesh resolution to reduce the element count; (iii) it controls element quality during mesh

smoothing; (iv) it reduces memory and CPU requirements for the solver by introducing mixed elements; (v) it reduces the running time using multi-cores; and (vi) it improves the overall reliability and portability of the code as it builds upon the ITK open-source, cross-platform system[1]. Chapter 2 presents the method along with evaluation results on multi-tissue data, as well as a comparison with four other popular image-to-mesh conversion techniques.

Image-to-mesh conversion can facilitate accurate computational modeling for FEM-based registration of Magnetic Resonance Imaging (MRI) in image-guided neurosurgery [81]. During surgery the brain deforms due to several factors such as cerebrospinal fluid leakage, intra-cranial pressure, gravity, tumor retraction, and resection [55, 134, 149]. Neuronavigation with non-rigid registration of preoperative MRI to intraoperative MRI allows the surgeon to view the location of surgical tools relative to the preoperative anatomical (MRI) or functional data (DT-MRI, fMRI), thus avoiding damage to eloquent areas during tumor resection [4, 45, 57, 122, 139, 151].

This thesis presents a biomechanical non-rigid registration framework utilizing multi-tissue mesh adaptation to automatically remove elements in the area of the resected tumor, thereby automatically handling deformation in the presence of resection. The proposed method augments a previous deformable registration method [120, 45]. The enhancements with mesh adaptation improve the accuracy of the registration by more than 5 times compared to rigid [98] and traditional physics-based deformable registration [120], and by more than 4 times compared to publicly available B-Spline interpolation methods [98].

The adaptive framework is parallelized for shared memory multiprocessor architectures. Acceptance in clinical practice requires that non-rigid registration be completed in the time constraints imposed by neurosurgery (e.g., 2-3 minutes) and without the cost of high-performance computing clusters. Therefore, we conduct performance analysis on a readily-available 12-core desktop system. The analysis shows that our algorithm could be applied, on average, in less than two minutes, achieving desirable speed for use in a clinical setting. Chapter 3 presents the adaptive non-rigid registration framework along with extensive evaluation results on 3-dimensional MRI data from thirty patients who underwent partial, total, and supra total glioma resection. Appendix A presents a customization of this framework to quantify brain shift during deep brain stimulation surgery using preoperative and intraoperative Computed Tomography (CT) scans.

[1] https://itk.org

Chapter 4 focuses on the discretization of CAD data. Finite element modeling using CAD data is an integral part of the design process in mechanical, civil, aerospace, and silicon engineering. For example, in aerospace, an unstructured mesh is necessary in Computational Fluid Dynamics (CFD) where Navier-Stokes calculations are performed to compute engine installation drag and lift loss due to an increased bypass ratio of turbofan engines for reducing the fuel consumption and engine noise [25]. A major difference between CAD and imaging is that the former defines the boundary of the object by means of constructive solid geometric primitives [144], or explicitly through a boundary discretization as a collection of patches [140]. On the other hand, imaging defines the boundary through an implicit function such that points in different regions of interest evaluate the function differently. This thesis is about volume dicretization, so it is assumed that the boundary of the domain is already meshed as a Piecewise Linear Complex (PLC) [140, 175].

CAD-based meshing can be challenging because the shape of the boundary patches (e.g., triangles) affects the shape of the volume elements (e.g., tetrahedra). Many applications require the preservation of a specific set of constraints (e.g., edges or faces) into the grid. The constraints can represent boundaries of the domain, an interface of a mesh partition, or other geometric features. Complex topologic modifications [73, 179, 97] and additional (*Steiner*) points might be necessary in order to preserve the constraints. The optimal locations and minimum number of Steiner points is still an open problem [165, 31]. All these facts make CAD-based volume meshing even more challenging.

This dissertation proposes a parallel unstructured mesh generator for efficient discretization of PLC domains in CFD modeling. Mesh generation codes typically improve the shape of the elements in a post-processing step called *mesh optimization* or *mesh quality improvement* [50, 71]. AFLR [132] is the only method we are aware of that optimizes the grid connectivity both during and after mesh generation (in a post-processing step) to obtain a maximum element quality. Our method not only optimizes the connectivity throughout the procedure, but it also utilizes multi-cores to perform parallel optimization, significantly improving execution time and end-user productivity without sacrificing element quality.

The proposed mesh generation engine is essentially a combination of Advancing Front type point placement [123, 127], direct element subdivision [132], and parallel

multi-threaded connectivity optimization schemes [171, 201]. The connectivity optimization is based on a speculative (optimistic) approach that has been proven to perform well on hardware-shared memory (i.e., single chip) [40, 70]. Topologic transformations [73, 179, 97] are employed to reconnect the grid. The parallel reconnection is based on: (i) data over-decomposition; (ii) atomic operations to avoid data races; and (iii) load-balancing to redistribute work-units among the threads.

Experimental evaluation on realistic aerospace configurations indicates that our code is stable and robust in that both sequential and parallel implementations generate grids of a comparable or higher element quality than state-of-the-art technology [132] currently incorporated in several commercial systems (i.e., DoD CREATE-MG Capstone, Lockheed Martin/DoD ACAD, Boeing MADCAP, MSU SolidMesh, and Altair HyperMesh). The proposed method improves end-user productivity, completing the mesh generation process (without post-improvement) up to 2.5 and 10 times faster compared to state-of-the-art code [132], when 1 and 12 hardware cores are utilized, respectively. Furthermore, it exhibits near-linear scalability when 24 cores are utilized to discretize the flow domain of a DLR-F6 Airbus aircraft (NASA's Common Research Model[2]) with approximately 1.5 billion elements. In future efforts, this mesh generator will be part of an Extreme-Scale Anisotropic Mesh Generation Environment [43] to meet industries expectations and NASA's CFD vision for 2030 [180].

[2]https://commonresearchmodel.larc.nasa.gov

CHAPTER 2

IMAGE-TO-MESH CONVERSION FOR ANATOMIC MODELING OF BIOMEDICAL APPLICATIONS

2.1 INTRODUCTION

Image-to-Mesh conversion algorithms are widely used for the quantitative analysis of patient-specific images using the Finite Element (FE) method. This has significant implications in many areas, such as image-guided therapy [185, 4, 58], interactive surgery simulation for training young clinicians [59], and endovascular flow diversion for aneurysm geometries [53, 54, 101]. The FE method is essential in modeling tissue deformation for these applications. An intrinsic difficulty of generating meshes from isosurface-based data (i.e. parametric surfaces/volumes, level-sets, segmented multi-labeled images) is the processing and recovery of the object geometry. General-purpose mesh generators (for solid and geometric modeling applications) expect that the object boundary is parameterized, i.e., it is defined by means of constructive solid geometry primitives, or explicitly defined, e.g., through the boundary discretization, as a collection of patches. In order to convert the isosurface data into a tetrahedral mesh, one can either (i) recover the parametrized object surface and follow up with a conventional mesh generation technique, or (ii) use a mesh generation method, which operates directly on this type of data.

This chapter presents a framework (CBC3D) to directly convert segmented multi-labeled image data into adaptive multi-tissue tetrahedral meshes that conform to the physical image boundaries. Image-to-Mesh conversion is an inherently difficult problem as it has to balance trade-offs between several criteria: (i) minimize the mesh size and maximize the element quality by utilizing proper mesh gradation and tissue-specific multi-resolution, (ii) without sacrificing geometric and topologic fidelity, (iii) while maintaining a smooth surface to reflect a certain degree of visual reality, and (iv) simultaneously, to allow the upstream numerical computations for collision detection and biomechanical Finite Element analysis to complete within the time constraints imposed by a surgical simulation.

The present procedure focuses on these common requirements and offers several advantages compared to previous implementations [65, 119]: (i) it improves the accuracy of

the geometric and topologic representation, (ii) it provides material-dependent mesh resolution to reduce the element count, (iii) it controls element quality during mesh smoothing, (iv) it reduces memory and CPU requirements for the solver by introducing mixed elements, (v) it reduces the running time using multi-cores, and (vi) it improves the overall reliability and portability of the code as it builds upon the ITK open-source, cross-platform system[1].

2.2 BACKGROUND

There are various meshing methods that operate on isosurface data. Some methods are based on a Delaunay refinement scheme [23, 159, 176], such as CGAL's 3D Mesh Engine[2]. The code prototype together with a first version of design and specifications is presented in [163]. CGAL first constructs a boundary mesh from the image and then a volume mesh given the surface mesh as an input. Before the refinement, a mechanism of protecting balls is set up on 1-dimensional features, if any, to ensure a fair representation of those features in the mesh. This mechanism also guarantees the termination of the refinement process independently of the input geometry. The refinement process is driven by criteria concerning either the size or shape of mesh cells and surface facets. It terminates when there are no more mesh cells or surface facets violating the criteria. The refinement is followed by a mesh optimization phase [33] to remove slivers and provide a good quality mesh.

A parallel 3D Delaunay meshing method (PODM) has been previously developed by our group [70]. PODM constructs the isosurface of the biological object with geometric and topological guarantees while also providing good shape elements. PODM implements a tightly-coupled, shared-memory, parallel speculative execution approach, using carefully designed contention managers, load balancing, synchronization and optimization schemes. The problem with the Delaunay refinement is that almost flat tetrahedra, called slivers, can survive. There are a number of optimization techniques to eliminate slivers [23, 33, 39], and some of them have been shown to produce very good dihedral angles in practice.

A large number of meshing methods are based on lattice space-tree (e.g., octree in 3D) decomposition, which provide adaptively-sized, high-quality elements for a number of medical applications [112, 26]. Usually, the mesh is tessellated by a regular background

[1] https://itk.org
[2] http://doc.cgal.org/latest/Mesh_3

mesh and the surface is recovered by finding the points where the mesh intersects the surface of the object. Then the mesh vertices are warped onto the model surface, or new vertices are inserted on the surface, and the mesh is locally modified. The achieved mesh quality depends on thresholds used for deciding whether to warp or insert a vertex. In some cases, a mesh decimation step is applied to an octree-based mesh to improve the element shape and/or modify the mesh size [38, 202]. In other cases, the element quality is improved in a post-processing step, by optimizing an objective function based on tetrahedral shape measures [72]. In [203] an efficient octree Dual Contouring (DC) method was presented to resolve topological ambiguity for multiple-material domains by analyzing a hybrid octree with both cubic and tetrahedral leaf cells.

Other approaches first generate an adaptive Body Centred Cubic (BCC) background lattice and then warp the surfaces of the lattice to the boundaries of the object using either a mass spring system, a finite element constitutive model, or an optimization scheme [143, 119]. In [75], an adaptive BCC lattice is created, the vertices outside the object are projected to the boundary, the Delaunay approach is used to tessellate the object, and a smoothing optimization step is applied. The method uses the lattice vertices only for estimating the initial positions of the mesh vertices. In [161], an advancing front technique generates the volumetric mesh using the vertices of a Face-Centred Cubic (FCC) lattice.

Previous work [65, 119] extended [143] to fairly regular multi-material geometries, and employing a physics-based mesh deformation scheme to warp the surfaces of a BCC mesh to the physical image boundaries. The deformation scheme took fidelity, smoothing, and quality into account. The advantage of using a physically based deformation scheme is that it can forecast how the mesh is likely to respond to the deformations it will experience during a simulation. Additional improvements for the reliable and faithful representation of multiple materials and their corresponding resolutions were presented in [59].

This thesis improves the accuracy of the geometric and topologic representation of the methods presented in [65, 119]. Lattice subdivision is performed using an Euclidian Distance Transform (EDT) [135] instead of a segmented image. In that way, smoother refinement is achieved within a threshold from the segmented image boundaries. To provide a more accurate topologic representation, the non-manifold voxel connectivities in the segmented image are eliminated. When the lattice subdivision is completed, the disconnected mesh regions or non-manifold mesh connectivities are detected, and the lattice is further subdivided (locally or globally) to resolve the image features.

The proposed method provides a material-dependent lattice subdivision to focus only on materials of interest, thus reducing the number of generated elements. Alternatively, global subdivision criteria can be specified. The badly shaped elements during a physics-based smoothing scheme are eliminated with heuristics and quality criteria such as a minimum dihedral angle or a scaled Jacobian metric. The trade-off between mesh fidelity and smoothing time is balanced with the extraction of different amounts of image features needed for smoothing. To reduce the number of vertices in the adaptive tetrahedral mesh, mixed elements (tetrahedra, pentahedra, and hexahedra) are introduced. The evaulation shows that a mixed mesh can reduce the number of vertices up to 30% without compromising the fidelity or smoothness of the tetrahedral mesh. Finally, this work utilizes parallel computing to improve the overall performance of the image-to-mesh conversion, allowing the processing of big data (e.g. Micro-CT Imaging of stents).

We compared our new CDC3D algorithm to four common image-to-mesh conversion methods: CGAL's 3D Mesh Engine[3] (v4.5.2), CLEAVER [26] (v1.5.4), Lattice-Derefinement (LD) [38], and PODM [70]. A single-material version of the CBC3D is available within 3D Slicer[4] package for visualization and image analysis. A previous multi-material version of the CBC3D is integrated within an interactive simulator for neurosurgical procedures involving brain Arteriovenous Malformations (AVM) developed in SOFA[5]; a framework for real-time medical simulations.

2.3 METHOD

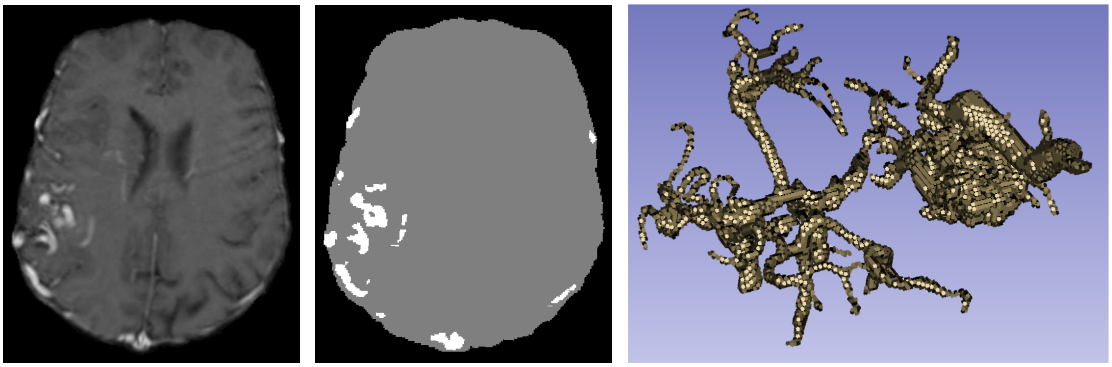
CBC3D uses a segmented multi-labeled image as input and creates an adaptive tetrahedral or mixed element mesh as an output. Subsection 2.3.1 describes the segmentation algorithm used to obtain the labeled image. Subsection 2.3.2 describes pre-processing algorithms used to improve the quality of the input image. Subsection 2.3.3 describes the generation of the adaptive BCC lattice. Subsection 2.3.4 describes the conversion of the adaptive tetrahedral mesh into a mixed element mesh. Subsection 2.3.5 describes mesh smoothing.

[3] http://doc.cgal.org/latest/Mesh_3
[4] https://www.slicer.org
[5] https://www.sofa-framework.org

2.3.1 SEGMENTATION

Before segmenting the vessels from the MR Imaging, a skull-stripping algorithm is applied to remove the skull bone from the image, using the Brain Extraction Tool (BET) [95]. The level set segmentation algorithm for the vessels structures uses the Vascular Modeling Toolkit (VMTK) [3]. Level sets are a kind of deformable model in which the deformable surface is represented by a 3D function whose contour at level zero is the surface in question. In order to extract the surface from the image, the output image is run through Marching Cubes [128] with level zero. A fast marching initialization is used, consisting of placing a set of seeds and a set of targets in the image. A front is then propagated from the seeds until the first target is met, at which point the region covered by the front is the initial deformable model. This type of initialization is effective when it is necessary to segment round objects such as aneurysms (e.g. by simply placing one seed at the center, and one target on the wall, the volume will be initialized). The output segmented image contains labels, each one representing a single tissue (Figure 3).

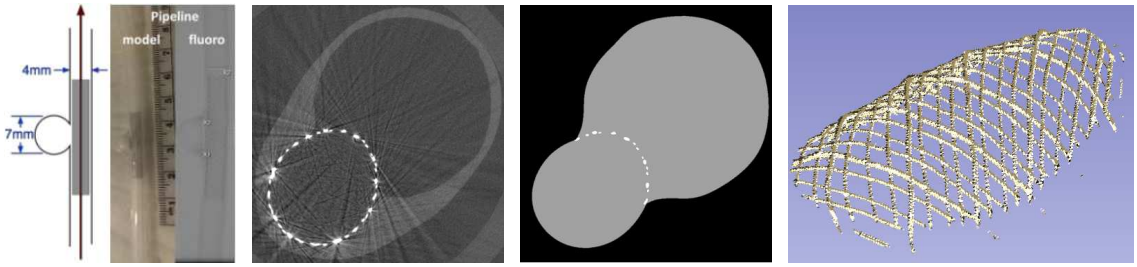
Materialise Mimics[6] software is used to perform image segmentation on the micro-CT data of commercially available neurovascular stents [53]. Two different labels are tagged on to the lumen of the sidewall aneurysm model and the flow diverter. During segmentation, the device is cropped to only retain the portion providing neck coverage of the aneurysm (Figure 4).



(a) (b) (c)

Figure 3: Brain MRI slice with AVM after skull stripping [59] (a) and after segmentation (b). (c) depicts a volume rendering of the segmented AVM. The segmented image has a spacing of $0.7 \times 0.7 \times 1.6$ mm^3 and a size of $320 \times 320 \times 100$ voxels^3.

[6] http://www.materialise.com/en/medical/software/mimics



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

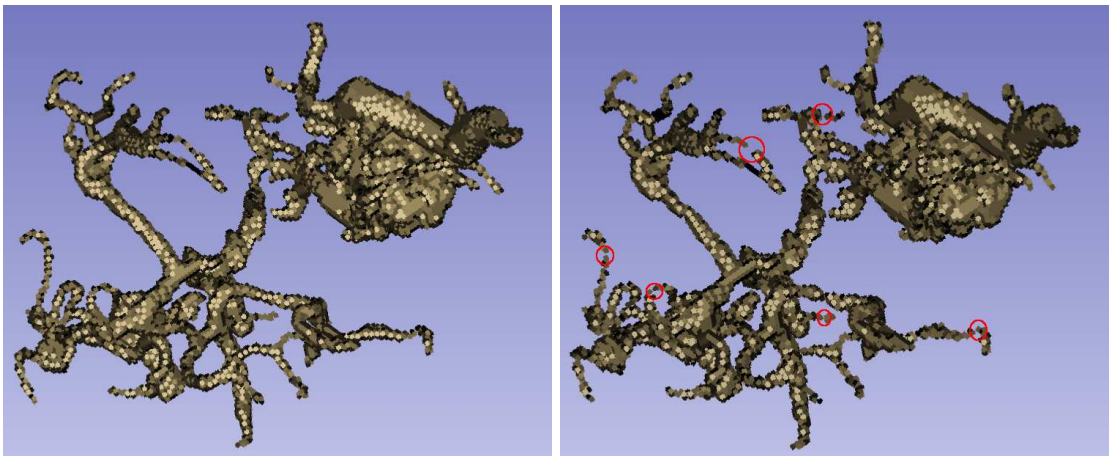
Figure 4: Raw micro-CT slice (b) for pipeline stent model [54] (a) and after segmentation with the device wires cropped (c). (d) depicts a volume rendering of the segmented stent. The segmented image has a spacing of $0.012 \times 0.012 \times 0.024$ mm^3 and a size of $1001 \times 1001 \times 4421$ voxels^3.

2.3.2 IMAGE PRE-PROCESSING

The previous work [65, 119] re-samples the segmented image to an isotropic unit spacing image to avoid the transformation of index to physical coordinates and vice versa. This approach is sufficient for fairly regular geometries and improves the speed of the mesh generation. However, in the case of complex data (e.g. AVM), image down-sampling can deteriorate the quality of the segmentation and may result in disconnected image regions or non-manifold connectivity (i.e., voxels that are connected to each other via an edge or a vertex). Figure 5 illustrates such an example. The current implementation does not perform image down-sampling.

Relabeling noisy voxels

A semi-automatic or manual segmentation of poor quality may contain isolated voxels that do not correspond to an anatomical image feature. CBC3D identifies two types of such voxels and relabels them based on their neighbors. The first type has a zero label (background), and all its neighbors have a non-zero label (Figure 6(a)). The second type has a non-zero label, and all its neighbors have a different label (Figure 6(b)). Not all neighbors necessarily have the same label. The neighbors are checked using a 26-neighborhood region (Figure 6(c)). After the noisy voxels are identified, they are relabeled to one of its neighbors' label. Either the first, the second, or both types are relabeled.



(a) Before down-sampling (spacing: $0.879 \times 0.879 \times 0.879$ mm^3)

(b) After down-sampling (spacing: $1 \times 1 \times 1$ mm^3)

Figure 5: Brain Arteriovenous Malformation (AVM) segmentation, before and after down-sampling. The red circles indicate the problematic regions after down-sampling (i.e., disconnected voxels or non-manifold voxel connectivity).

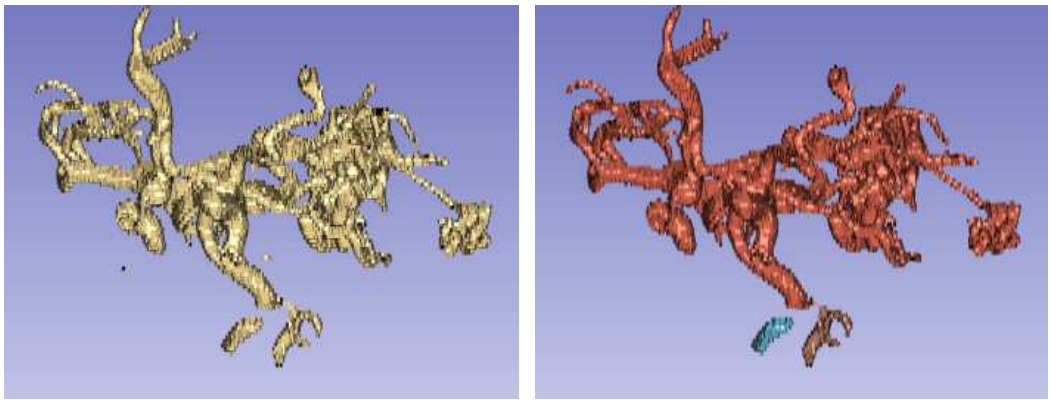
Relabeling disconnected regions

CBC3D controls the level of refinement in each material by using a global or a material-specific fidelity (subsection 2.3.3). Complex segmentations (i.e, tangled bundles of abnormal and arbitrary thin blood vessels connecting arteries and veins in the brain) may contain materials with multiple disconnected regions. CBC3D implements a relabeling algorithm to handle each of those regions as a different material. Therefore, the present method allows a more localized refinement which leads to a lower element count.

An ITK connected threshold image filter is employed to detect the disconnected regions in each material. A seed is first computed based on the label of the material, and then small regions on the image with the same label are merged iteratively to compute a connected region. A face connectivity is used for merging. Assume that S_i is the set of voxels in material i, and the filter computes a set of voxels S_{ij} in a region j of material i. If $|S_i| = |S_{ij}|$ (where $|*|$ is the number of voxels in $*$), then material i is a single region. Otherwise, material i consists of disconnected regions. If $\frac{|S_i|-|S_{ij}|}{|S_i|} < s_{tol}$ then region j is too small compared to region i, hence it is relabeled to zero (background). Otherwise, region j is relabeled to a new label. A tolerance $s_{tol} = 10^{-4}$ is used to detect the small disconnected regions. Figure 7 depicts an example before and after the relabeling process.

(a) (b) (c)

Figure 6: Candidate isolated voxels for relabeling (a)-(b) and 26-neighborhood region (c).



(a) Before relabeling (b) After relabeling

Figure 7: Relabeling an AVM segmented image with disconnected vessels. Before processing, the image contains one material (yellow) with five disconnected regions (a). After processing, the image contains three materials (red, cyan, and brown) but each of those is a single region (a). Two small disconnected regions are relabeled to a background value.

Eliminating non-manifold voxel connectivity

A segmentation with poor resolution may contain non-background voxels which, are connected through a single edge or a vertex (Figure 9(a)). This type of connectivity can lead to a non-manifold mesh (i.e., tetrahedra which are connected through a mesh edge or a mesh vertex), especially when refinement is inadequate to resolve the small image features. A non-manifold mesh deteriorates the solution accuracy in blood flow simulations within stented arterial segments or AVM surgical simulations (Figures 3-4).

CBC3D incorporates relabeling operations with specific templates in order to eliminate the non-manifold voxel connectivities. First, all vertex-to-vertex connectivities are detected (Figure 8(a)), and eliminated by randomly selecting one of the six templates (Figures 8(b)-8(g)). All templates result in a face-to-face connectivity; however, each template relabels different voxels. When all vertex-to-vertex connectivities are eliminated, the edge-to-edge connectivities are detected (Figure 8(h)) and eliminated by randomly selecting one of the two templates in Figures 8(i)-8(j). Similarly, the two templates relabel different voxels. The two steps repeat until all non-manifold connectivities are eliminated. Randomness is introduced in the selection of those templates to achieve convergence in the case of relabeling neighbor clusters. Figure 9 depicts an example of an AVM segmented image before and after elimination.

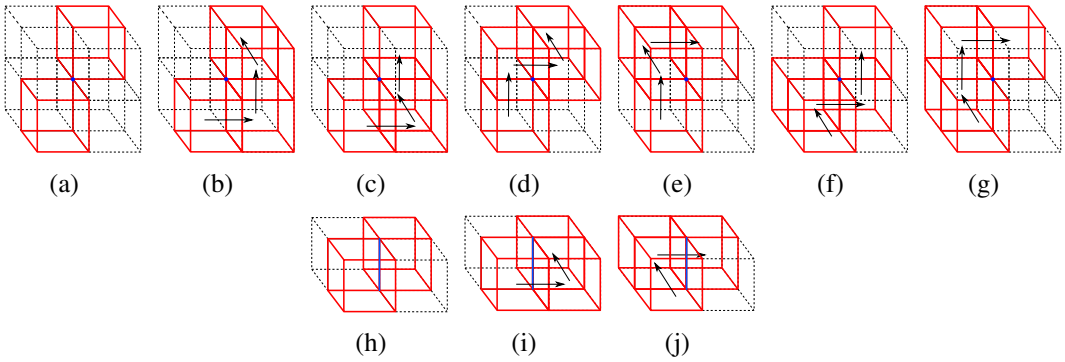
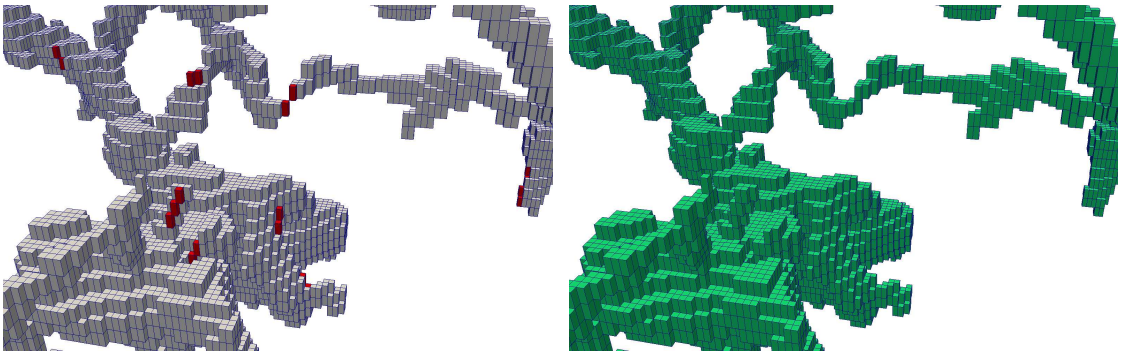


Figure 8: Templates to eliminate a vertex-to-vertex connectivity (a) or an edge-to edge connectivity (h) in a segmented labeled image. In the case of a vertex-to-vertex connectivity, one of the six templates is randomly selected to relabel two voxels within a cluster of eight voxels (b)-(g). In the case of an edge-to-edge connectivity, one of the two templates is randomly selected to relabel a single voxel within a cluster of four voxels (i)-(j). The arrows illustrate the path of the transformation via face connected voxels after relabeling.

2.3.3 ADAPTIVE LATTICE REFINEMENT

BCC lattice construction

Initially the image is discretized with a regular Body Centred Cubic (BCC) lattice. This structured lattice is generated as a result of two interlaced lattices (Figure 10). The first lattice is created by setting its vertices with an input-defined distance between them (BCC_size parameter in Table 3), then the edges connecting those vertices are added,



(a) Before relabeling (b) After relabeling

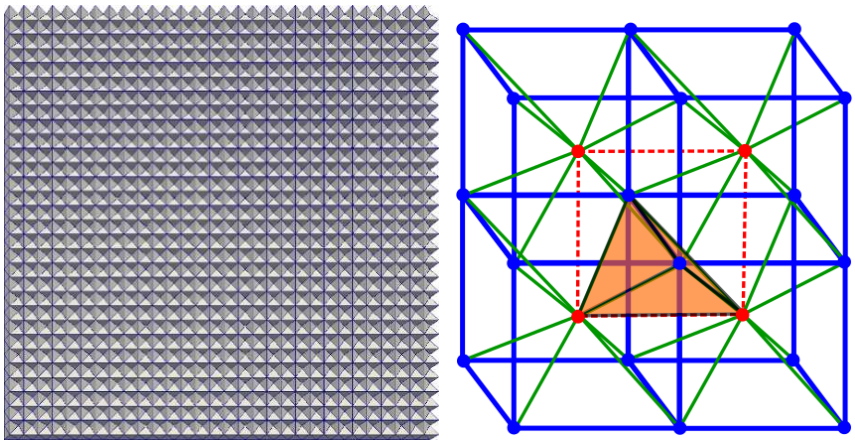
Figure 9: AVM anisotropic segmented image $(0.7 \times 0.7 \times 1.6\ mm^3)$ before (a) and after (b) relabeling to eliminate the non-manifold voxel connectivity. The voxels which are connected via an edge or a vertex are depicted with red color.

consequently creating the tetrahedral elements. The elements of the second lattice are created with consideration to the first lattice so that their vertices correspond to the centroids of the initial lattice cells. As a final step, the tetrahedra located completely outside of the object are discarded. The resulting elements are of the best quality possible (the minimum dihedral angle is $60°$) with the regular space tiling [75].

Adaptive refinement

With the purpose of subdividing the elements, a metric has to be considered. The current implementation uses an Euclidean Distance Transform (EDT) to decide whether an element needs to be subdivided (Figure 11). An EDT provides more information compared to a segmented image, and thus it conducts a smoother refinement nearby the material boundaries. In the presence of n materials, n EDTs will be calculated in order to exclusively subdivide the elements within each material. ITK's Signed Maurer Distance Map Image Filter [135] is employed to compute the EDT in parallel and reduce the running time.

The lattice is refined using red (regular) - green (irregular) templates (Figure 12). An element is subdivided if the EDT in which the centroid of the element belongs to changes sign in at least one of the four element vertices. Initially, all tetrahedra are marked as red. A red tetrahedron is always subdivided into eight children (1:8 regular refinement), and each child is marked red, as shown in Figure 12(a). There are three choices for the internal edge of the red tetrahedron. If the shortest one is selected, the resulting eight child tetrahedra



(a) Cut section (b) Lattice detail

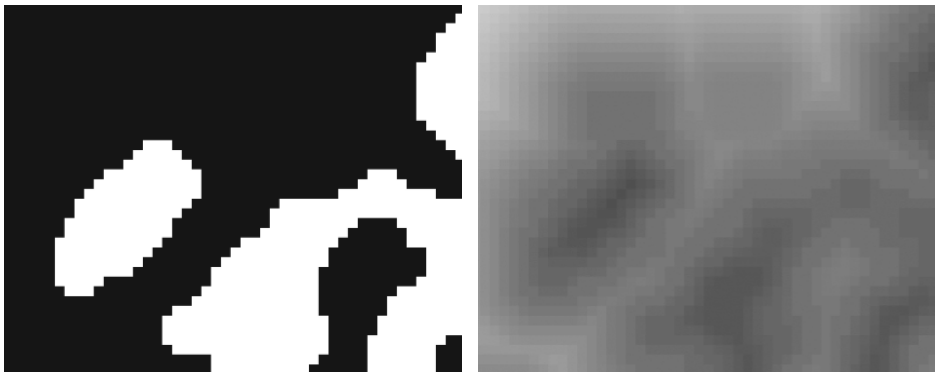
Figure 10: Uniform Body-Centered Cubic (BCC) lattice. The green edges lace the two lattices together. Each vertex is surrounded by 14 edges and 24 tetrahedra.

are exactly the same as the parent except the size is one half of the parent's size. The red subdivision will lead to T-junctions at the newly created edge midpoints where neighboring tetrahedra are not refined to the same level. To remove the T-junctions, a green (irregular) subdivision which, includes the three cases depicted in Figures 12(b)-12(d), is performed.

To quantitatively evaluate the similarity between a mesh that corresponds to a single material (sub-mesh) and the image region of this material (sub-region), S_1 is defined as the set of all voxels in the sub-mesh, S_2 as the set of all voxels in the sub-region, and $S_1 \cap S_2$ as the point-set shared by the sub-mesh and the sub-region (common region). Ratio $F_1 = \frac{|S_1 \cap S_2|}{|S_1|}$ corresponds to the similarity between the common region and the sub-mesh, and ratio $F_2 = \frac{|S_1 \cap S_2|}{|S_2|}$ corresponds to the similarity between the common region and the sub-region. The refinement criterion is defined as: Refine the sub-mesh if $F > F_1$ or $F > F_2$, where $F \in (0, 1]$ is a global input fidelity or a material-specific fidelity (Table 3). The higher the input fidelity the finer the mesh nearby the boundaries. The advantage of a material-specific fidelity compared to a global fidelity is that the former allows a more localized refinement in the materials of interest, thereby reducing total element count.

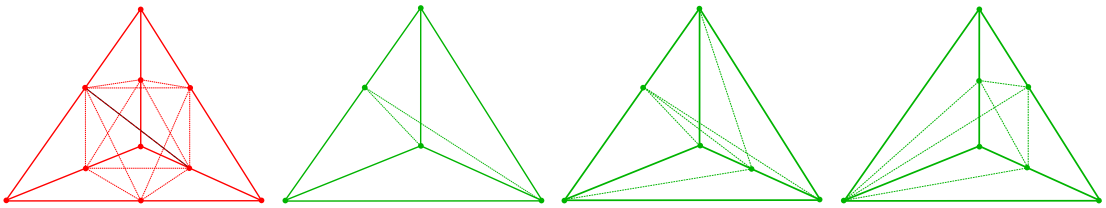
Candidate mesh selection

Looking forward into the future steps of the algorithm, for a surface to be suitable for smoothing, each non-background element is allowed to be connected via at most one



(a) Segmented image (b) EDT of (a)

Figure 11: Euclidean Distance Transform (EDT) computed from a labeled image. The EDT calculates the minimum distance of a voxel in the image from its closest material boundary. Voxels inside a material have a positive distance value, voxels outside a material have a negative distance value and voxels on the boundary have a zero distance value.

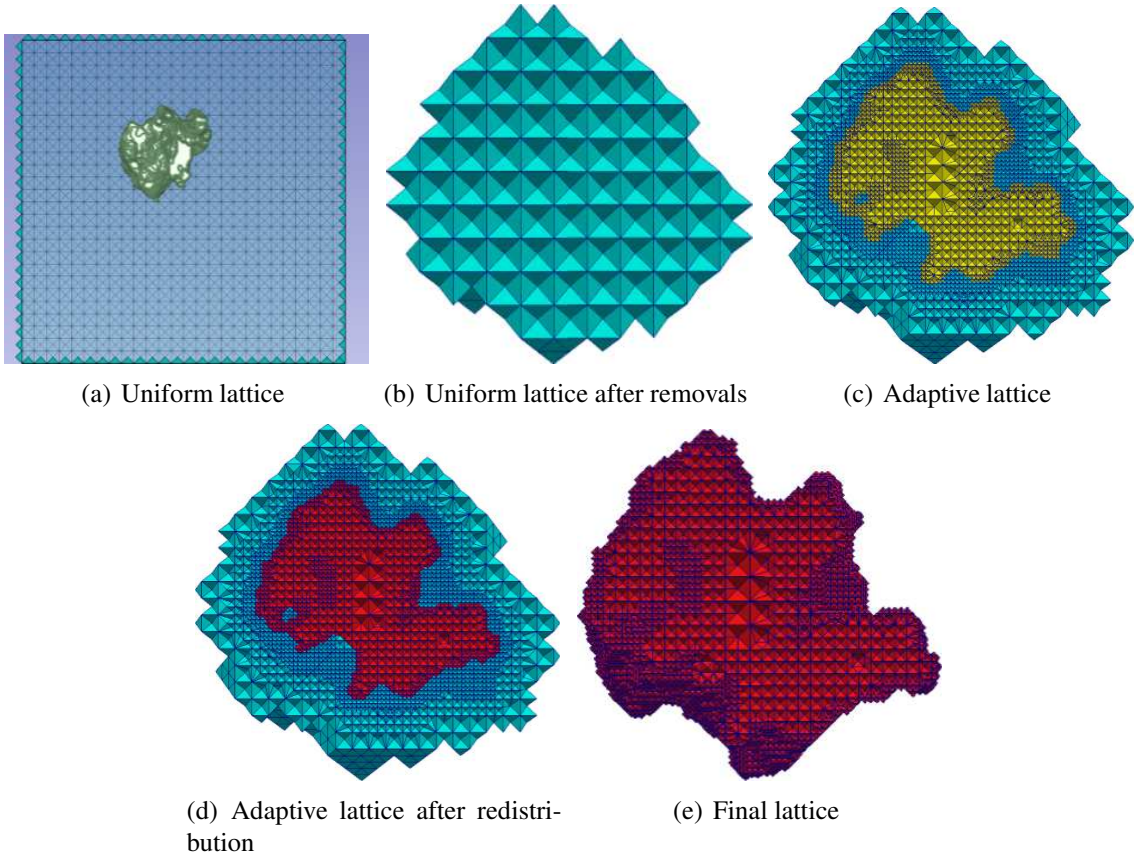


(a) 1 : 8 (b) 1 : 2 (c) 1 : 4 (d) 1 : 4

Figure 12: Red-Green templates for lattice subdivision. 8, 2, and 4 is the number of tetrahedra after subdivision.

face with a background element. Otherwise the elements on the surface can be easily distorted or inverted during smoothing. To avoid this type of connectivity, the labels of those problematic surface elements and of the surrounding elements are redistributed. Every tetrahedron is examined by observing the labels of its four adjacent tetrahedra. If at least three of the adjacent tetrahedra have the same label as the examined tetrahedron, then there is no need for relabeling. However, if there is more than one adjacent tetrahedron with a different label, then the examined tetrahedron needs to be relabeled to one of its neighboring labels. Between these labels, the one chosen for relabeling the tetrahedron depends on whether a considered label has been already examined and finalized in the previous examinations. So, if a neighboring label has not been fully inspected yet, the tetrahedron

is relabeled to this one. At the end of the relabeling process, the background elements are discarded and the final mesh is obtained. Figure 13 depicts the basic steps of the adaptive mesh genaration.



(a) Uniform lattice (b) Uniform lattice after removals (c) Adaptive lattice

(d) Adaptive lattice after redistribution (e) Final lattice

Figure 13: Pipeline of the BCC lattice construction and adaptive refinement.

Mesh topological checks

The goal is to obtain a manifold topology, meaning that every tetrahedron must be connected to one another through a face. After the lattice subdivision is completed (i.e., all input fidelities are satified) and the candidate mesh is selected, the mesh topology is examined to ensure the absence of non-manifold (i.e. connected via an edge or a vertex) tetrahedral connections or disconnected tetrahedra in each sub-mesh.

A mesh connected threshold filter is developed to detect the non-manifold topology or the disconnected tetrahedral regions in each sub-mesh. An arbitrary seed tetrahedron is first computed based on the label of each sub-mesh, and then neighbor tetrahedra are merged iteratively to compute a connected mesh region. A vertex connectivity is used for

merging. The non-manifold edges and the non-manifold vertices are identified by checking the labels of the neighbor elements before merging.

When the connected threshold filter is completed, the number of tetrahedra in the connected mesh region is compared with the number of tetrahedra in the sub-mesh. If they are equal, the sub-mesh is a single connected region. However, if non-manifold topologies exist, then the tetrahedra surrounding all non-manifold vertices and all non-manifold edges are marked and they are locally subdivided in the next refinement iteration. Otherwise, the refinement of the sub-mesh is completed.

If the number of tetrahedra in the connected mesh region is not equal to the number of tetrahedra in the sub-mesh, then the sub-mesh contains disconnected regions. In this case, all disconnected regions are first checked for relabeling. A disconnected region is relabeled if its volume is too small $(< 1\%)$ compared to the volume of the sub-mesh. Relabeling is performed using a label of a neighbor region (background or not). Relabeling is cancelled if it produces a non-valid manifold connectivity. If all disconnected regions are eliminated and all connectivities are manifold, then the refinement of this sub-mesh is completed. If disconnected regions exist, then the whole sub-mesh is globally refined in the next iteration. Finally, if non-manifold topologies exist, then the tetrahedra surrounding the non-manifold vertices and the non-manifold edges are marked and locally subdivided in the next refinement iteration.

The mesh topological checks should be performed only if the non-manifold voxel connectivities have been previously eliminated (subsubsection 2.3.2). Also, mesh topological checks can be time consuming. Therefore, they should employed only when the application requires a manifold mesh or when the image contains small features which are harder to resolve (e.g., vessels or stents).

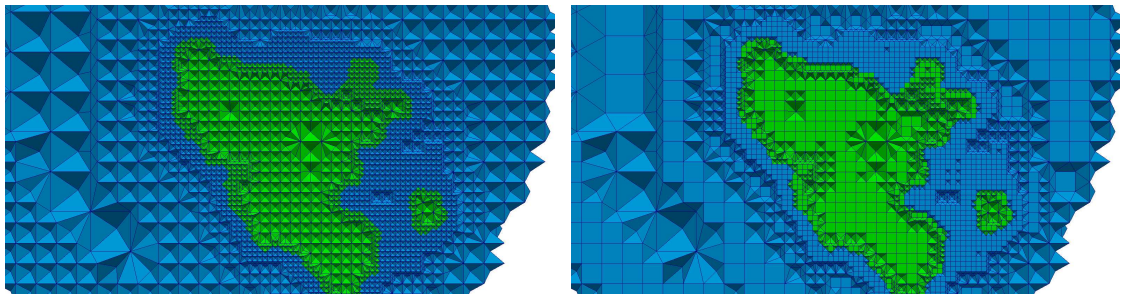
2.3.4 MIXED ELEMENT MESH

A mixed mesh typically contains tetrahedra, pentahedra, and hexahedra. Experimental evaluation in this study showed that a mixed mesh may contain up to 30% fewer vertices compared to a tetrahedral mesh of the same input, without compromising the fidelity. Therefore, it can reduce the subsequent memory and CPU requirements for the solver without any loss of accuracy.

CBC3D generates a mixed mesh from an adaptive tetrahedral lattice by merging clusters of tetrahedra into hexahedra. A valid transition between the tetrahedra and the hexahedra is guaranteed with the use of prismatic elements (i.e., pyramids). Merging is performed

only within the homogeneous regions where the lattice is uniform. Therefore, the topology of the generated surfaces or interfaces between materials is preserved.

In the BCC lattice, the cardinality of a vertex in a uniform region is always 14, meaning that each vertex has 14 edges to check in the process of merging (Figure 10(b)). Beginning from the generated adaptive BCC mesh, the inner tetrahedral vertices are located to start the transformations. For every vertex, its adjacent tetrahedra constitute a polyhedron. To transform each one of these polyhedra to hexahedra, their inner tetrahedra must be red, according to the red-green refinement and their labels need to be the same. If indeed all the tetrahedra of the polyhedron are of the same label and none of them is a green one, then for each adjacent orthogonal edge of the vertex (Figure 10(b)), its four attached tetrahedra are computed. Their four orthogonal edges form the quadrilateral face of the hexahedron. However, if an attached tetrahedron between these four is green or has a different label than the others, then a transition pyramid is created. This element has its base perpendicular on the orthogonal edge and by being adjacent to the hexahedral face, its vertices and edges are calculated. After creating each hexahedron, the tetrahedra inside the hexahedron are removed and the mesh topology is updated. Figure 14 compares qualitatively a tetrahedral and a mixed mesh of the same input. The smoothing algorithm takes place with respect to three different element types: the tetrahedron, the hexahedron, and the pyramid.



(a) Tetrahedral mesh (b) Mixed element mesh

Figure 14: Adaptive BCC lattice before (a) and after (b) it is converted into a mixed element mesh.

2.3.5 MESH SMOOTHING

Red-green templates guarantee the high quality of the adaptive mesh. Indeed, the worse minimum dihedral angle is 30° and the worse maximum dihedral angle is 116.5°. The surface of this mesh can be relatively bumpy, so it may not be suitable for applications such as AVM surgical simulations or CFD modeling of brain aneurysms/stents for endovascular

flow diversion. For example, a surgical simulation requires a smooth surface so that the mesh will reflect a certain degree of visual reality. Bumpy surface can deteriorate the accuracy of a CFD solution. The accuracy of the CFD solutions may be improved further by incorporating a boundary layer mesh [61].

A smoothing approach that deforms the mesh surfaces to their corresponding image boundaries by minimizing an energy function was presented in [119]. The present study improves [119] in several ways. First, the new implementation is built upon the ITK toolkit, therefore improving the overall reliability and portability of the software. Second, it employs heuristics and quality control to eliminate the highly distorted elements during deformation. Third, it allows meshes with mixed elements to be smoothed. Fourth, it controls the trade-off between smoothing time and mesh fidelity. Finally, it reduces the smoothing time using parallel computing.

This study formulates the smoothing problem as an energy minimization problem represented by the objective function

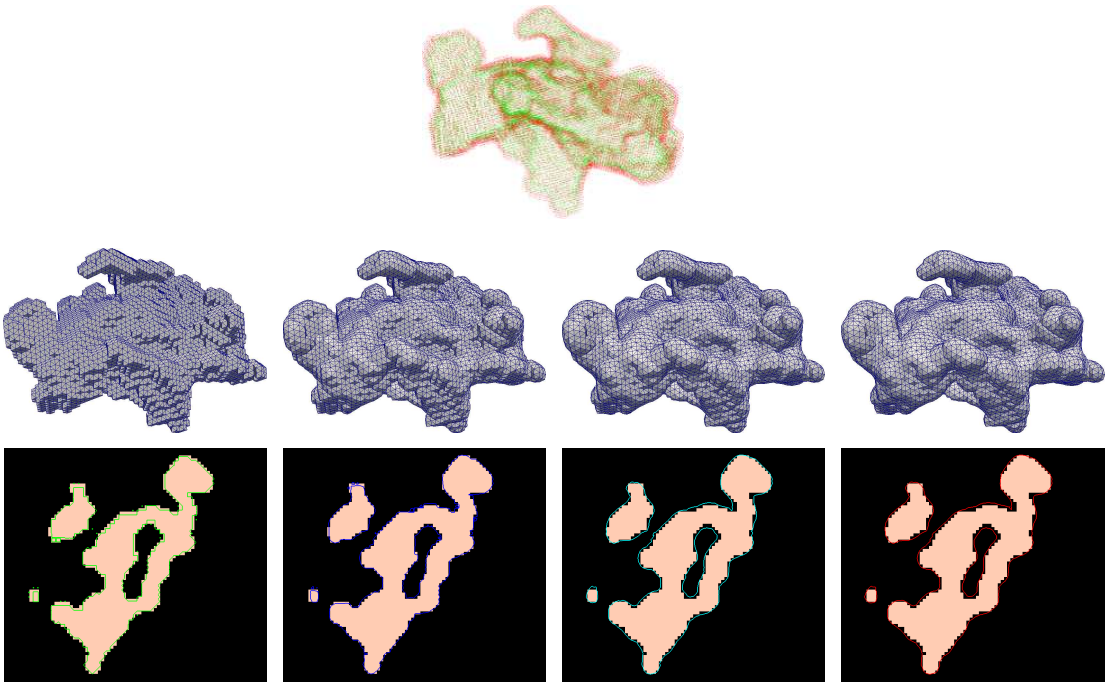
$$W = U^T K U + (HU - D)^T (HU - D) \tag{1}$$

U is the unknown mesh displacement vector, K is the mesh stiffness matrix, H is a linear interpolation matrix, and D is the vector of correspondences computed between two point-sets: a source and a traget point set. The source point-set contains the coordinates of the surface/interface vertices of the mesh to be smoothed. The target point-set contains the coordinates of the voxels on the surface/interface of the materials in the image. The K matrix depends on the element type and the mechanical properties of the brain model. First, the stiffness matrix of each element (e.g., tetrahedron, pyramid, hexahedron) is calculated in each integration point using Gaussian quadrature. The global matrix K is then assembled from the individual stiffness matrices [15].

To minimize (1), the gradient with respect to U must be zero, leading to a linear system of equations:

$$(K + H^T H)U = H^T D \tag{2}$$

An iterative solver is employed [107] to solve the system. A linear assumption is made for the materials and the displacements of the biomechanical model. Once U has been found, the mesh is updated by adding the computed displacements to the current coordinates of the vertices, and the procedure is repeated until the iterations reach the maximum number. Figure 15 illustrates an example of the procedure.



(a) $HD_0 = 1.79$ mm (b) $HD_3 = 1.31$ mm (c) $HD_7 = 0.96$ mm (d) $HD_{10} = 0.91$ mm

Figure 15: Nidus mesh during smoothing with 10 iterations. The figure on the top depicts the extracted source (green) and target (red) points used for smoothing. Each column depicts the smoothened mesh and an intersection between the mesh surface and the image plane at iterations $i = 0, 3, 7, 10$ (from left to right). HD_i denotes the mesh fidelity in terms of a Hausdorff Distance metric, at iteration i. The smaller the HD value, the higher the fidelity. As the number of iterations advances, the mesh exhibits a smoother surface.

Extraction of source and target points using multi-cores

The source and the target points are extracted from the mesh to be smoothed and the segmented image, respectively. The source points are the vertices on the mesh surface or a the interface between sub-meshes of different materials. The target points are the physical centers of the voxels on the segmented image boundaries.

The extraction is performed before smoothing is started. For the extraction of the source points, the mesh is first decomposed into k sub-meshes so that the number of elements in each sub-mesh is approximately the same (k is the number of threads). Each thread extracts points from a single sub-mesh. The labels of the elements surrounding each source point are stored in a label-set. This is because the smoothing essentially registers pairs of source and target points with same label-sets. For the extraction of the target points, the segmented image is first partitioned into a number of k sub-regions using

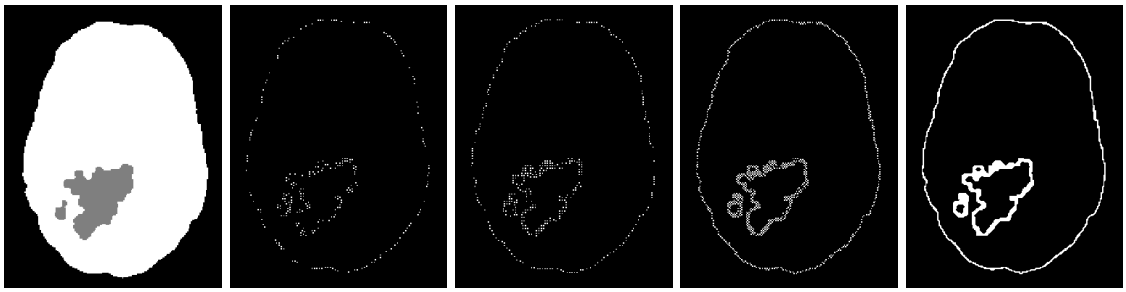
an ITK threaded image region partitioner filter. Points are extracted from the segmented boundaries of each sub-region with the help of EDT's (EDT's are previously calculated at the refinement step). Additionally, the labels within a 26-neighborhood region (Figure 6(c)) around the target point are stored. Figure 15 depicts the extracted source and target points from a Nidus tetrahedral mesh and image, respectively.

Quality control

For each source point, an average displacement vector is calculated from the corresponding target points within a 3D region around the source point [119]. To improve element quality after each deformation iteration, the size of this 3D region is limited by a local element size, i.e., the average length of the element edges surrounding the source point. Then vector D is assembled from the calculated displacement vector of each source point. After each deformation iteration, a quality metric (i.e., minimum dihedral angle for tetrahedra or scaled Jacobian for hexahedra and pyramids [110]) is computed for each element. The elements that do not satisfy a minimum quality (e.g., 5° for dihedral angle or 0.2 for scaled Jacobian) are marked, and the displacement vectors of the source points surrounded by at least one marked element are scaled with a factor of 0.2. D is assembled again and the system in (2) is solved. If the quality metric is not satisfied after three consecutive attempts, the smoothing stops and the procedure recovers the mesh coordinates of the previous deformation iteration.

Adjustable number of target points

The number of source points is fixed during smoothing because the mesh remains topologically the same; however, the number of target points can be adjusted to improve the accuracy of the displacement vector of a source point. Connectivity patterns are used to control the number of extracted target points. The connectivity patterns prohibit the selection of points that are too close to each other. A "vertex" (26-connectivity), an "edge" (18-connectivity), or a "face" (6-connectivity) pattern avoids the selection of neighboring voxels connected via a "vertex", an "edge" or a "face", respectively. The "no" option disables the connectivity patterns, hence maximizing the number of selected points. Figure 16 depicts the extracted target points using the available patterns. Table 1 presents qualitative results on mesh smoothing using those patterns. The higher the number of target points, the more accurate but slower the computation. For the experiments in this study, a "face" pattern is employed to balance the trade-off between accuracy and speed.



(a) Segmentation (b) “Vertex” (c) “Edge” (d) “Face” (e) “No”

Figure 16: Extracted target points from a brain-nidus segmented image using the available connectivity patterns. Each pattern results to a different number of points. The number of points for “Vertex”, “Edge”, “Face”, and “No” pattern is 21510, 26387, 47306, and 91906, respectively. For simplicity, the points in one volumetric slice are depicted.

Table 1: Performance of smoothing of a nidus geometry (Figure 15) using the available non-connectivity patterns. HD is a Hausdorff Distance metric. HD_{10} corresponds to the mesh fidelity after a deformation step of 10 iterations. $HD_0 = 1.79$ mm is the mesh fidelity before the deformation. The smaller the HD value, the higher the fidelity. The experiment conducted on a machine with an Intel i7-2600@3.40 GHz CPU, and 16 GB of RAM.

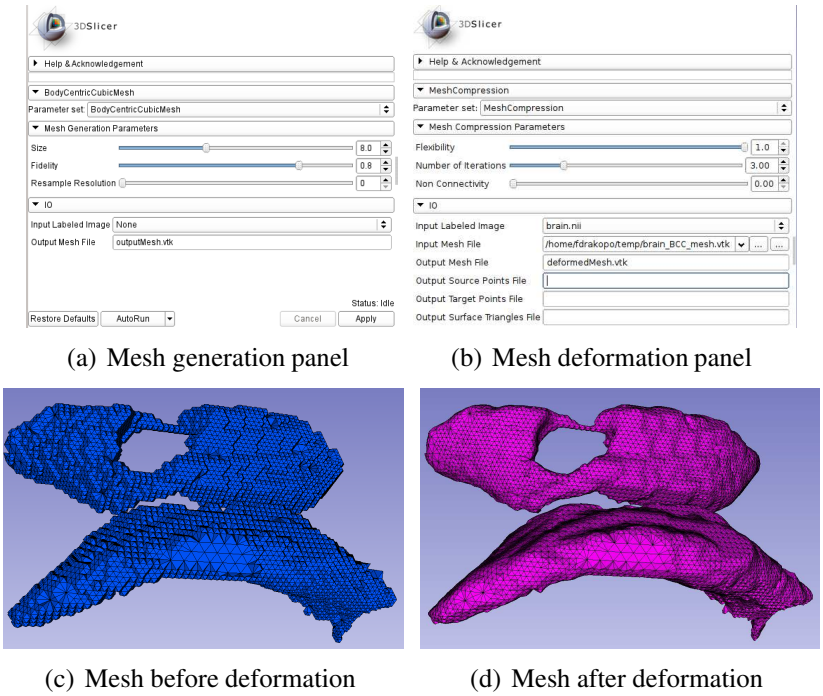
Pattern	#Target Points	HD_{10} (mm)	Time (sec)
“Vertex”	2770	0.97	21.01
“Edge”	3498	0.91	22.57
“Face”	6213	0.88	26.19
“No”	11755	0.84	35.27

2.4 SOFTWARE IMPLEMENTATION

The CBC3D software is available as: (i) a stand-alone ITK library (multi-tissue version), (ii) a 3D Slicer extension (single-tissue version), or (iii) a SOFA plugin within an interactive simulator (multi-tissue version with limited features).

2.4.1 3D SLICER EXTENSION

3D Slicer is a free, open-source C++ software application widely used in the medical computing research community. It builds upon VTK and ITK for visualization and image processing, respectively. Slicer facilitates the clinical translation of medical computing research ideas by providing developers with an easy-to-use framework for image analysis



(a) Mesh generation panel (b) Mesh deformation panel

(c) Mesh before deformation (d) Mesh after deformation

Figure 17: CBC3D extension in 3D Slicer. (a), (b) depict the extension's GUI. (c), (d) depict a ventricle mesh generated with the extension.

and visualization. It also provides a template framework that allows for the creation of customized and sophisticated user interfaces. The CBC3D meshing method is available as an extension in 3D Slicer (Figure 17). Slicer supports different types of extensions that can be built outside of the source tree and bundle together one or more modules.

2.4.2 SOFA-BASED INTERACTIVE SIMULATOR

One of the purposes of the CBC3D software is to be used to generate anatomically correct AVM models for surgical simulations. Interactive simulations involving deforming solids are widely used in physically-based simulations. In particular, accurate and efficient simulation tools are critical for the design and development of surgical simulations due to the real-time computation requirements and physics fidelity. However, because of the complexity of the problem, these simulations are computationally intensive, making interactive and real-time constraints a very difficult problem. Broadly, the main components for most interactive medical simulators require knowledge in the following areas: bio-mechanical and anatomical modeling, collision detection, haptics and visualization. A

virtual anatomical model constitutes a precise computerized description of a human organ that is commonly generated from medical images, like MRI or CT. In this case, it describes an AVM pathology containing draining vessels, feeding arteries, and the nidus of vessels bundles as well as surrounding structures. The brain matter interacts with vessels, fluid, and possibly other rigid objects, so a very important aspect of the simulator is the ability to provide anatomically correct models of the structures involved in the virtual surgical procedure. At the same time, the anatomical model should contain the minimal amount of elements needed to accurately describe the small vessel features. In this study, an interactive simulator for neurosurgical procedures is developed involving brain vasculature. The simulator builds upon the SOFA open-source framework for real-time medical simulation [64]. The simulator's development framework consists of FEM biomechanical anatomical modeling and volumetric meshing of vascular structures, coupling collision detection and response with haptic feedback, with GPU-based implementations enabling real-time simulation.

SOFA Plugin. The CBC3D method is also available as a plugin in SOFA. The simulator uses this plugin to generate a tetrahedral mesh of the AVM pathology. The visual or collision model is created by mapping a surface triangle topology to the surface of the tetrahedral mesh. The plugin mechanism exposes SOFA's API and allows us to use all the necessary features and data structures without the need to tightly integrate the CBC3D with the rest of the SOFA library.

Collision Detection. SOFA's current capabilities in terms of reliable collision detection and response computation are limited. It only performs discrete collision, i.e., it checks for collisions between the mesh at fixed time steps. Furthermore, it has limited capabilities detecting self intersections. This study employs a continuous collision detection method (Self-CCD) developed in Gamma Lab at the University of North Carolina. This algorithm performs inter and intra objects queries using various culling techniques (i.e. representative triangles [49], connectivity [188], and other non-penetration filters [187]) to speed up computation and robustness.

FEM Simulator. Tissue response is handled by a corotational tetrahedral FEM formulation [66]. In future efforts, a hybrid FEM approach will be integrated to increase performance (Figure 18). The hybrid FEM increases the performance of the tissue response calculation by utilizing two formulations on a single mesh: a linear formulation for fast computation, albeit less accurate, and a non-linear formulation for more realistic simulations near the interaction area. It then couples these two formulations seamlessly

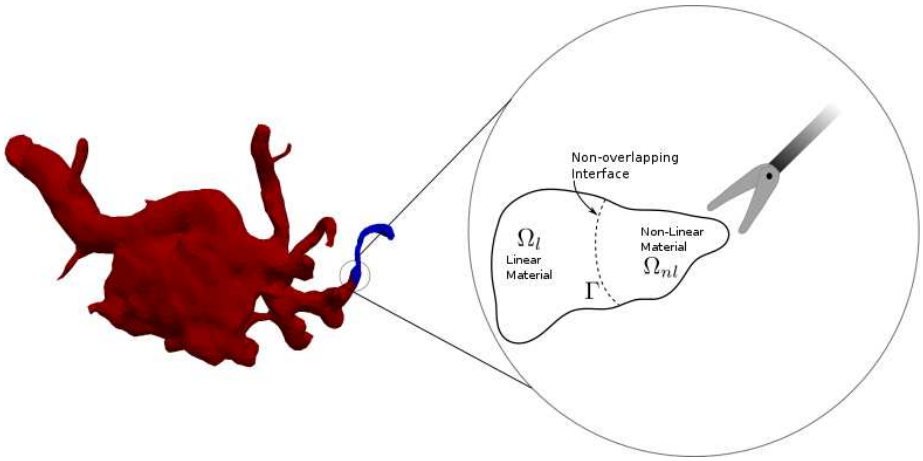


Figure 18: Conceptual depiction of the hybrid FEM method.

by updating the feedback forces at the interface of the two formulations [6]. With AVM surgery simulation involving various tissues with different physical properties, the hybrid FEM promises improvements in performance on the physics engine by up to 50%. Figure 19 shows preliminary results of the interactive simulator.

2.5 EVALUATION RESULTS

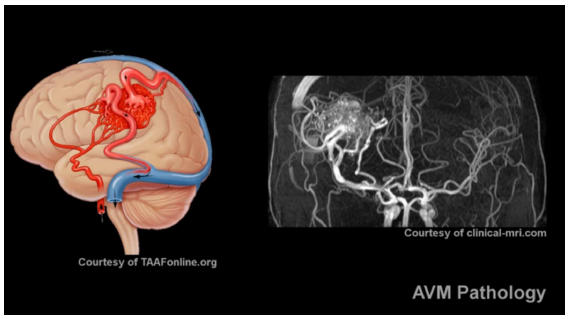
The CBC3D software is evaluated on four segmented images. Table 2 lists the image data. Cases 1 and 4 are obtained from the Neurosurgery Department at Stony Brook University. Cases 2 and 3 are obtained from Kitware[7]. In case 4, the volume image is obtained after combining 4182 bitmap slices using an ITK Tile Image Filter.

Table 2: Segmented imaging data for experimental evaluation.

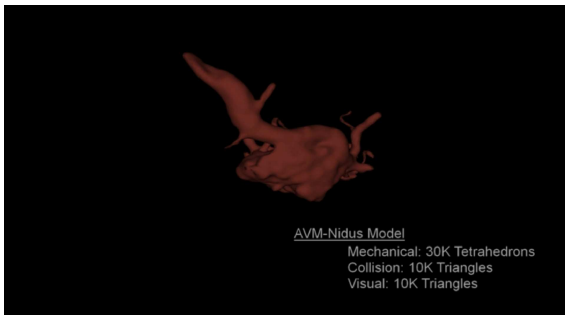
Case	Type	Materials	Image Spacing (mm^3)	Image Size (voxels^3)
1	Isotropic	Aneurysm	$1.00 \times 1.00 \times 1.00$	$512 \times 512 \times 508$
2	Anisotropic	Brain-Tumor	$0.48 \times 0.48 \times 1.00$	$384 \times 512 \times 176$
3	Anisotropic	Brain-AVM	$0.70 \times 0.70 \times 1.60$	$320 \times 320 \times 100$
4	Anisotropic	Lumen-LVIS Stent	$0.012 \times 0.012 \times 0.024$	$1001 \times 1001 \times 4182$

CBC3D is compared with four image-to-mesh conversion codes: CGAL's 3D Mesh Engine[8] (v4.5.2), CLEAVER [26] (v1.5.4), Lattice-Derefinement (LD) [38], and

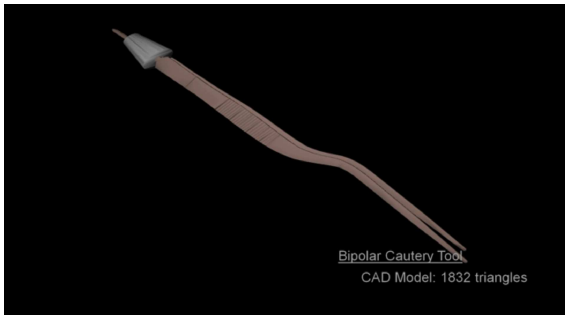
[7]https://www.kitware.com
[8]http://doc.cgal.org/latest/Mesh_3



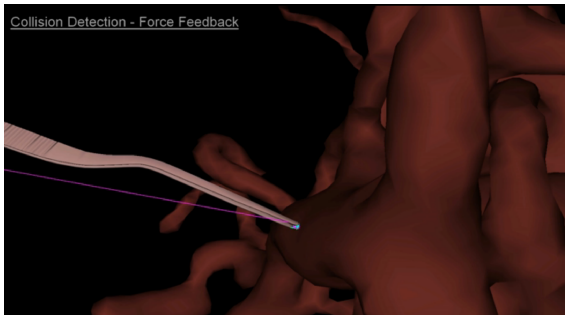
(a) AVM pathology



(b) AVM model



(c) Cautery tool



(d) Collision detection-interaction

Figure 19: Interactive simulator for neurosurgical procedures involving brain vasculature, in SOFA. The simulator uses a CBC3D plugin to generate a volumetric mesh of the vascular structures.

PODM [70]. CGAL, and CLEAVER are open-source codes. LD and PODM are developed by CRTC[9]. CBC3D, CLEAVER, and LD are lattice-based methods. CGAL and PODM are Delaunay-based methods.

CGAL's 3D Mesh Engine is based on a Delaunay refinement scheme [176]. CGAL first constructs a boundary mesh from the image and then a volume mesh given the surface mesh as an input. The refinement is followed by a mesh optimization phase [33] to remove slivers and provide a good quality mesh.

PODM is a parallel, Delaunay-based, image-to-mesh conversion algorithm with quality and fidelity guarantees achieved by dynamic point insertions and removals [70]. PODM implements a tightly-coupled, shared-memory, parallel speculative execution approach. It employs carefully designed contention managers, load balancing, synchronization and optimizations schemes. Parameter $\delta > 0$ specifies the size of the mesh (Table 3). The smaller the δ, the higher the element count.

[9]https://crtc.cs.odu.edu

Cleaver is a multi-material lattice-based meshing method [26]. Initially, it constructs a high-quality BCC lattice that covers the input image. Next, it locally warps or cuts the lattice so it will conform to multi-material arbitrary surfaces. The violation parameters $\alpha_{short}, \alpha_{long}$ decide the trade-off between snapping/warping and stencil cleaving; therefore, they implicitly control the size of the mesh. In this evaluation, the default violation parameters are used (Table 3), thus a worse case minimum dihedral angle of 2.76° and worst case maximum dihedral angle of 175.42° is achieved.

LD is an image-to-mesh conversion that allows for guaranteed bounds on the smallest dihedral angle and on the distance between the boundaries of the mesh and the boundaries of the materials [38]. The number of tetrahedra in the mesh is as small as possible provided that the quality, and the distance requirements are satisfied. LD initially constructs a voxelized mesh with very high quality and fidelity. A postprocessing decimation step follows to coarsen the mesh to a much lower number of elements while at all times maintaining the required fidelity and quality bounds. The highest possible fidelity and a minimum dihedral angle of 15° is specified for all the experiments. Table 3 lists the parameters for the evaluation.

Table 3: Input parameters for experimental evaluation.

Method	Parameter	Value	Description
CBC3D	$lattice_{sp}$	10 mm (Cases 1-3) 0.15 mm (Case 4)	lattice spacing
	F	0.95	fidelity
	$connectivity$	"no"	pattern for target point extraction
	N_{iter}	5 (Cases 1-3) 7 (Case 4)	number of smoothing iterations
CGAL	$facet\_angle$	30°	lower angle bound for surface facets
	$facet\_size$	2	radii upper-bound for Delaunay balls
	$facet\_distance$	2	face distance upper bound
	$cell\_radius\_edge\_ratio$	1.5	radius-edge ratio upper bound
	$cell\_size$	2.5 (Cases 1-2) 1.5 (Case 3) 0.2 (Case 4)	circumradii upper-bound
CLEAVER	α_{short}	0.357	diagonal edge threshold for edge-cuts
	α_{long}	0.203	axis-aligned threshold for edge-cuts
LD	$i2m$	0	image-to-mesh distance
	$m2i$	0	mesh-to-image distance
	$angle$	15°	minimum dihedral angle
PODM	δ	1.2 (Cases 1-2) 0.6 (Case 3) 0.02 (Case 4)	element size

Figures 20 and 21 depict cuts of the generated meshes. Cleaver exhibits the most rapid element gradation among all the methods. CGAL, CLEAVER, and LD fail to generate a mesh for Lumen-LVIS stent (Table 2). CGAL's image reader cannot read the input image. CLEAVER terminates the program with a message: "Cleaver Tet Mesher terminated with an unknown exception". LD throws an instance of std::bad_alloc because it exceeds the physical memory of the system (757 GB). For comparison purposes, CBC3D and PODM allocate 354 GB and 141 GB, respectively.

Table 4 reports quantitative results on element count and element quality. The lattice-based methods (CBC3D, CLEAVER, LD) result in larger meshes compared to Delaunay-based methods (CGAL, PODM) mainly because the templates impose stricter rules at element subdivision. Nevertheless, the largest mesh in this study is generated by PODM (15.85 M) because a small element size ($\delta = 0.02$) is specified to resolve the LVIS stent. Figure 22 compares the meshes of the LVIS stent. PODM does not adequately resolve the features of the stent, hence the mesh appears to be disconnected (Figure 22(b)).

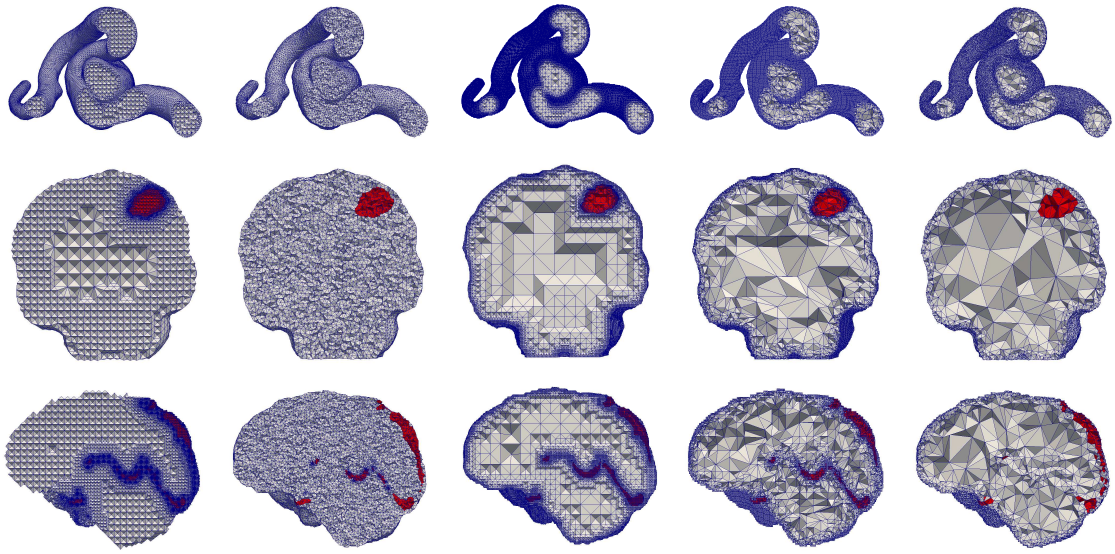
CLEAVER and LD generate meshes with high quality elements, although no experimental results are available for case 4 (Table 4). LD imposes a quality bound in the generated mesh, i.e., minimum dihedral angle of 15° by adjusting the element count appropriately. CBC3D produces meshes of well shaped elements. Smoothing does not significantly affect case 3, hence the angle extrema are similar to those in the adaptive lattice. The Delaunay-based methods generate meshes of reasonably well shaped elements.

Table 4: Evaluation results on element count and element quality. The minimum and maximum dihedral angles are reported in degrees $\in (0^\circ, 180^\circ)$. The larger the minimum angle and the smaller the maximum angle, the higher the quality.

Case	#Tetrahedra					Dihedral angle (min, max)				
	CBC3D	CGAL	CLEAVER	LD	PODM	CBC3D	CGAL	CLEAVER	LD	PODM
1	272K	300K	4.33M	776K	370K	$(5.07^\circ, 171.73^\circ)$	$(12.04^\circ, 162.23^\circ)$	$(25.06^\circ, 126.94^\circ)$	$(15.00^\circ, 171.05^\circ)$	$(4.54^\circ, 170.13^\circ)$
2	578K	517K	3.20M	2.54M	244K	$(8.89^\circ, 166.67^\circ)$	$(12.00^\circ, 162.73^\circ)$	$(11.34^\circ, 153.15^\circ)$	$(15.00^\circ, 171.86^\circ)$	$(4.90^\circ, 170.26^\circ)$
3	5.05M	1.68M	3.59M	1.48M	1.03M	$(29.94^\circ, 116.67^\circ)$	$(1.41^\circ, 176.76^\circ)$	$(5.81^\circ, 157.66^\circ)$	$(15.00^\circ, 170.55^\circ)$	$(4.40^\circ, 170.24^\circ)$
4	12.98M	-	-	-	15.85M	$(4.95^\circ, 173.10^\circ)$	-	-	-	$(2.23^\circ, 176.44^\circ)$

Figures 23-26 illustrate the element angle distribution of the meshes. The Delaunay-based methods exhibit a Gaussian distribution of dihedral angles. The lattice-based methods exhibit a different distribution due to a more structured connectivity. For example, in CLEAVER, about about 41% of the dihedral angles are between $60^\circ - 65^\circ$, and about 26% are $90^\circ - 95^\circ$. In CBC3D, about 50% of the dihedral angles are between $55^\circ - 65^\circ$, and about 20% are between $85^\circ - 95^\circ$. In LD, about 15% of the angles are between $45^\circ - 50^\circ$, and about 20% are between $90^\circ - 95^\circ$.

The mesh fidelity is qualitatively evaluated on AVM data (case 3). For this purpose,

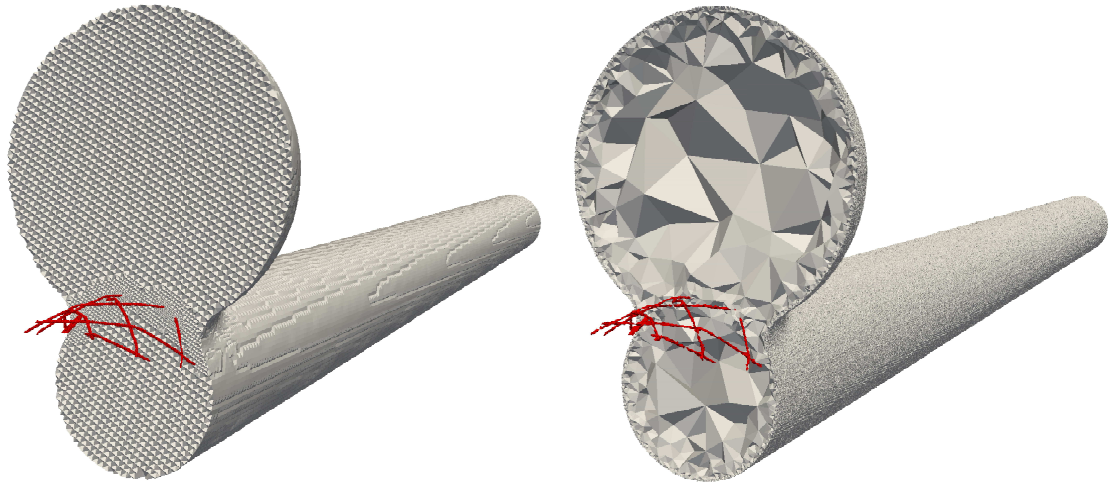


(a) CBC3D (b) CGAL (c) CLEAVER (d) LD (e) PODM

Figure 20: Cuts of the generated tetrahedral meshes. The top, middle, and bottom row correspond to cases 1, 2, and 3, respectively. Each column depicts meshes which are generated with a single method. Identical cut section planes are used for all the meshes in a single case. The growth from small to large elements varies among the methods. The quality of these meshes is evaluated using a min/max dihedral angle metric and an element angle distribution in 5-deg increments.

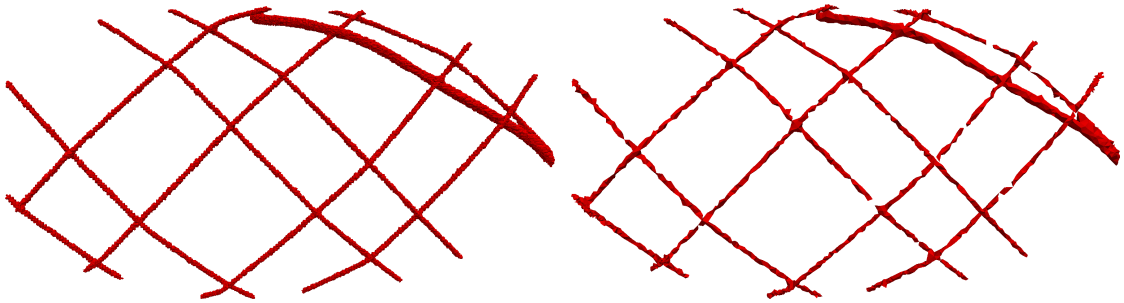
the AVM mesh is first extracted from the multi-material mesh and then superimposed on the AVM segmentation (Figure 27). The closer the mesh surface to the boundary of the segmented AVM image, the higher the fidelity. The LD method achieves a high fidelity because it completely resolves the vessels (it creates a voxelized mesh surface). Nevertheless, Figure 27(e) indicates a small shift in the output mesh in relation to the input image, most likely due to image resampling. CLEAVER resolves most of the vessel structures, but the generated mesh is noticeably shifted (Figure 27(d)). CBC3D achieves a satisfactory fidelity. CBC3D's mesh topological checks are turned off to avoid further red-green subdivisions, keeping the element count low.

Quantitative evaluation on mesh fidelity is performed using a two-sided Hausdorff Distance metric: $\mathrm{HD} = \max\{\mathrm{HD}_{\mathrm{I}\to\mathrm{M}}, \mathrm{HD}_{\mathrm{M}\to\mathrm{I}}\}$, where $\mathrm{HD}_{\mathrm{I}\to\mathrm{M}}$ is the value of the metric from the image to the mesh, and $\mathrm{HD}_{\mathrm{M}\to\mathrm{I}}$ is the value of the metric from the mesh to the image. The lower the HD error, the higher the fidelity. HD is computed between two point sets. The first point set contains the coordinates of the vertices located on the mesh surface. The second point set contains the coordinates of the voxels located on the boundaries of the



(a) CBC3D (b) PODM

Figure 21: Cuts of the generated tetrahedral meshes of the Lumen-LVIS Stent.



(a) CBC3D (b) PODM

Figure 22: Extracted mesh of the LVIS stent.

segmented material (i.e., where the EDT value is zero). HD is computed for each material and the maximum value is reported. An open-source implementation is employed to calculate the HD metric [47]. Table 5 reports the HD values and the end-to-end running time for mesh generation. PODM is a parallel multi-threaded code. CBC3D is sequential but computes the EDTs, the source and the target points in parallel. The rest of the codes are sequential. Figure 28 depicts plots of the results.

It should be noted that the error in Table 5 is relative to the image spacing. An error approximately 3-4 times the spacing may be satisfactory, depending on the application. LD achieves the highest fidelity among all the methods, but no experimental results are

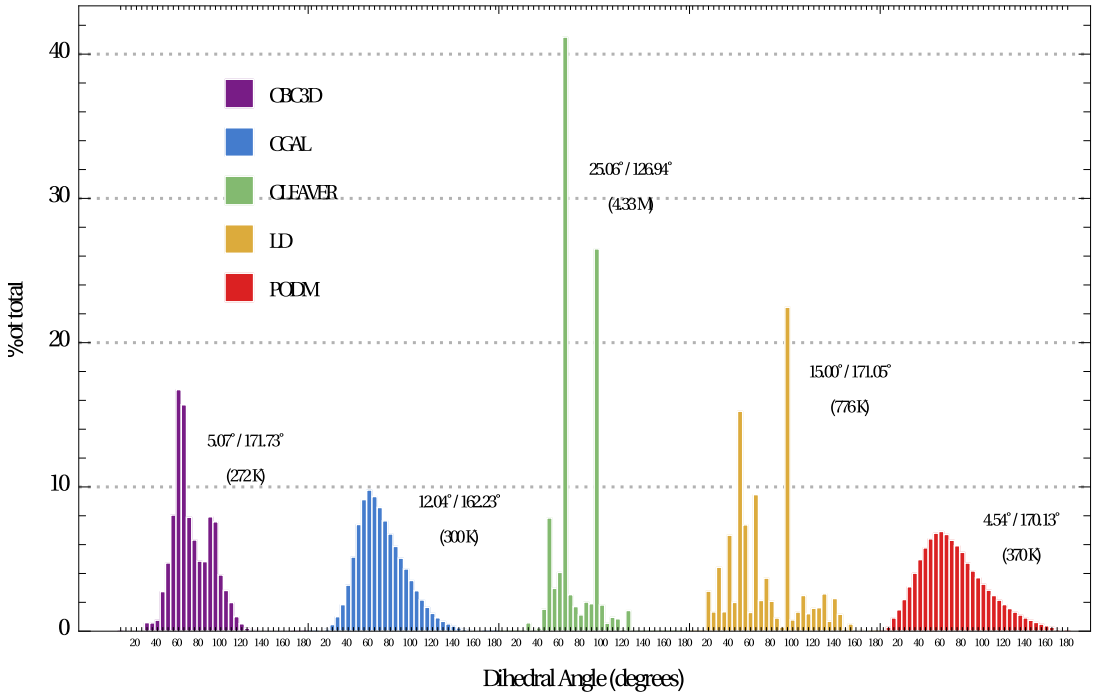


Figure 23: Element angle distribution (in 5-deg increments) of Cavernous Aneurysm meshes. The min/max dihedral angles and the element count are reported for each method.

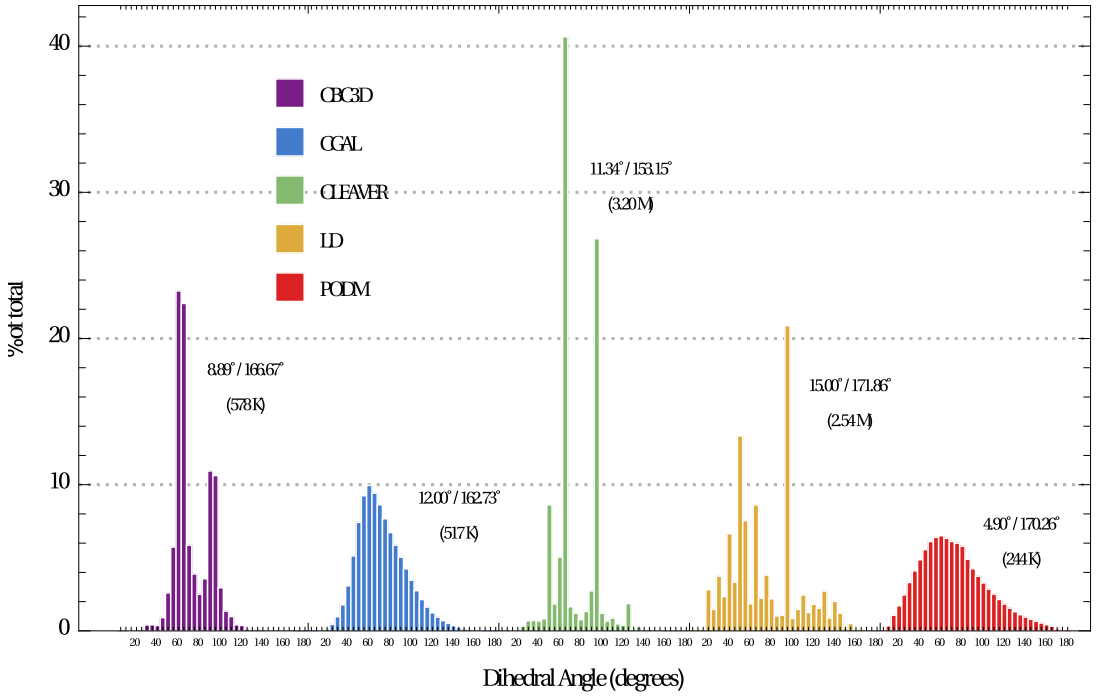


Figure 24: Element angle distribution (in 5-deg increments) of Brain-Tumor meshes. The min/max dihedral angles and the element count are reported for each method.

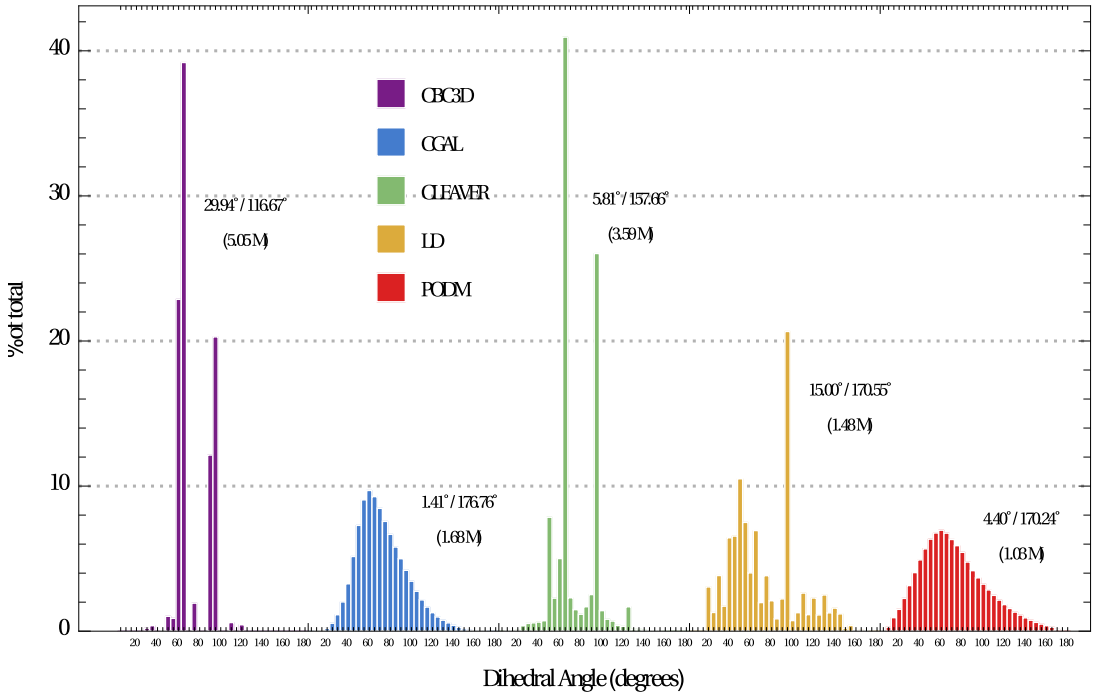


Figure 25: Element angle distribution (in 5-deg increments) of Brain-AVM meshes. The min/max dihedral angles and the element count are reported for each method.

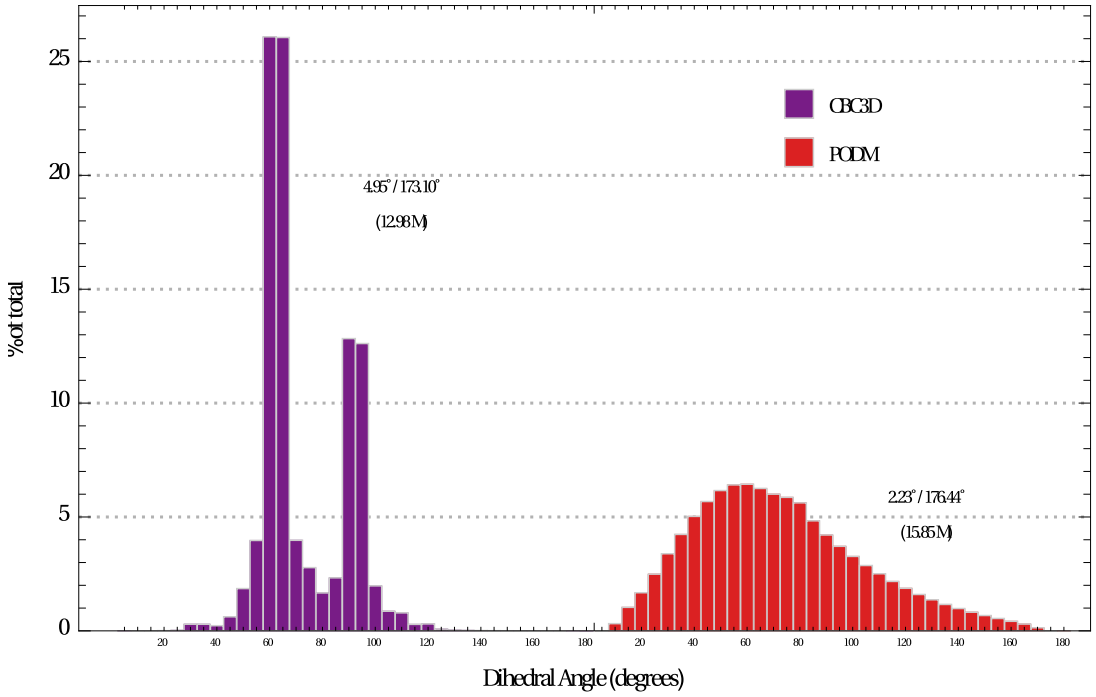
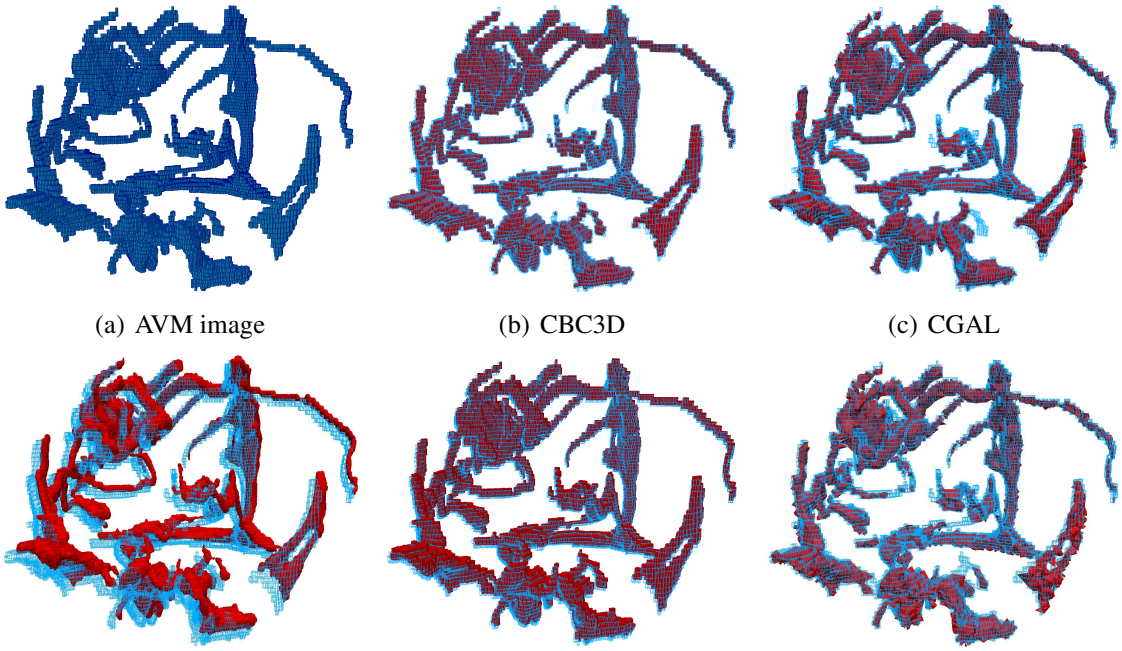


Figure 26: Element angle distribution (in 5-deg increments) of Lumen-LVIS stent meshes. The min/max dihedral angles and the element count are reported for each method.



(a) AVM image (b) CBC3D (c) CGAL

(d) CLEAVER (e) LD (f) PODM

Figure 27: Qualitative evaluation on fidelity of AVM mesh. Figures (b)-(f) depict the AVM mesh (red) superimposed on the AVM segmentation (blue). The closer the mesh surface to the boundary of the segmented material, the higher the fidelity.

available for case 4. PODM and CBC3D exhibit a reasonably good fidelity.

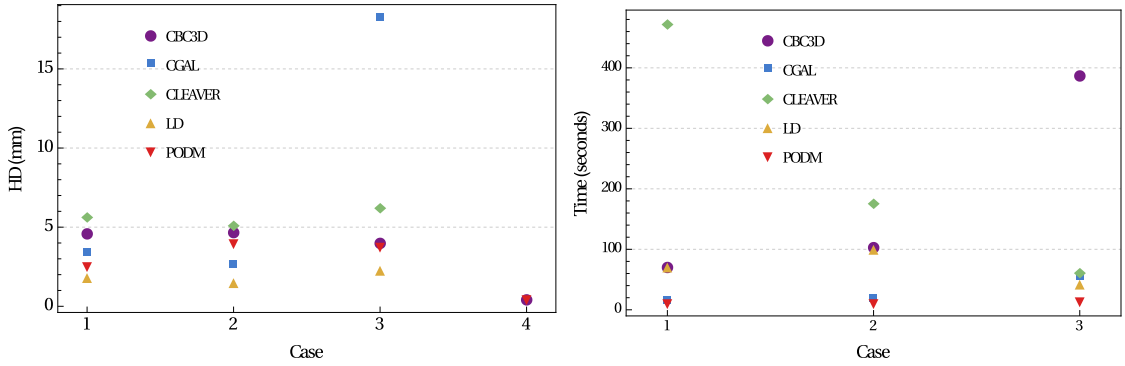
A fine mesh that accurately conforms to a segmented image typically has a voxelized appearance. However, the voxelized segmentation is itself inaccurate. The true surface is smooth and a binary segmentation does not capture this fact. On the other hand, a smooth surface mesh is easier to obtain with a lower fidelity. Therefore, fidelity must be compromised in favour of a more realistic appereance. Applications such as interactive surgical simulations, require volume meshes with smooth surfaces. Additionally, a bumpy surface can deteriorate the accuracy of a CFD solution.

Figure 29 compares the surface meshes of a Cavernous Aneurysm. CBC3D exhibits the smoothest surface among all the methods. CGAL creates a relatively smooth surface as it follows a two-step approach; it first reconstructs the surface and then generates a volume mesh given the surface as an input. CLEAVER and LD generate voxelized surfaces. PODM generates a bumpy surface (Figure 29).

CBC3D's tetrahedral meshes are converted to mixed meshes. Figures 30 and 31 depict the results. A mixed mesh reduces the number of mesh vertices, as well as the subsequent

Table 5: Results on mesh fidelity and end-to-end running time. Fidelity is calculated using a two sided Hausdorff Distance metric. The lower the HD value, the higher the fidelity. The experiments for cases 1-2 are performed on a DELL workstation with 8 hardware cores Intel(R) Core(TM) i7-2600 CPU @ 3.40 GHz, and 16 GB RAM. The experiments for cases 3-4 are performed on a DELL workstation with 24 hardware cores Intel(R) Xeon(R) CPU E5-2697v2@2.70 GHz, and 757 GB RAM. The time for writing the mesh is not included.

Case	HD (mm)					Mesh Generation Time (seconds)				
	CBC3D	CGAL	CLEAVER	LD	PODM	CBC3D	CGAL	CLEAVER	LD	PODM
1	4.51	3.26	5.56	1.73	2.42	68.24 (8T)	12.33	470.22	68.44	7.86 (8T)
2	4.59	2.51	5.03	1.41	3.87	101.03 (8T)	14.92	173.85	97.76	7.93 (8T)
3	3.91	18.09	6.14	2.19	3.65	384.86 (24T)	51.59	59.13	40.19	10.98 (24T)
4	0.34	-	-	-	0.33	3932.98 (24T)	-	-	-	750.02 (24T)



(a) HD metric

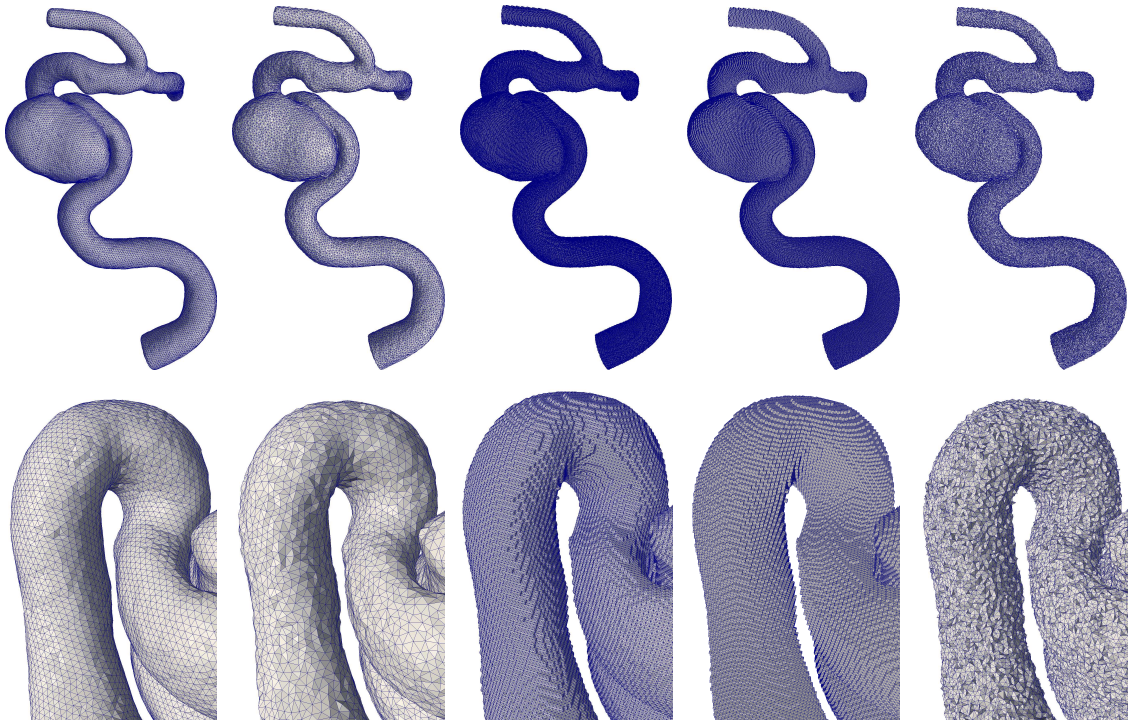
(b) Mesh generation time

Figure 28: Plots of the results in Table 5. (b) does not include case 4 (3932.98, and 750.02 seconds for CBC3D and PODM, respectively).

memory and CPU requirements for the solver without any loss of accuracy. The number of vertices is reduced by 2.35%, 22.28%, 23.40%, and 30.22% for cases 1, 2, 3, and 4, respectively. The mixed meshes exhibit similar qualities, and smooth surfaces to the adaptive tetrahedral meshes.

2.6 CONCLUSIONS AND FUTURE WORK

An adaptive, image-to-mesh conversion method is presented. This method initially discretizes a segmented labeled image with a uniform BCC lattice of high quality tetrahedra. Then, it employs red-green templates to subdive the lattice near the segmented boundaries, therefore improving mesh conformity. Finally, the generated surfaces are deformed to their



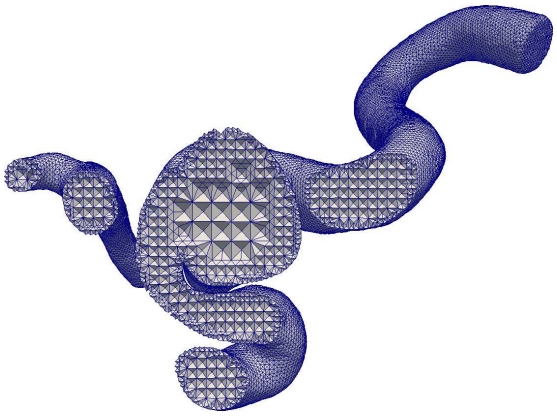
(a) CBC3D (b) CGAL (c) CLEAVER (d) LD (e) PODM

Figure 29: Comparison between the surface meshes of case 1 (Cavernous Aneurysm). Each column corresponds to a single method. The bottom row depicts a closer view of the surface. Among the methods, only CBC3D approximates the voxelized segmentation with a smooth surface that reflects a certain degree of visual reality.

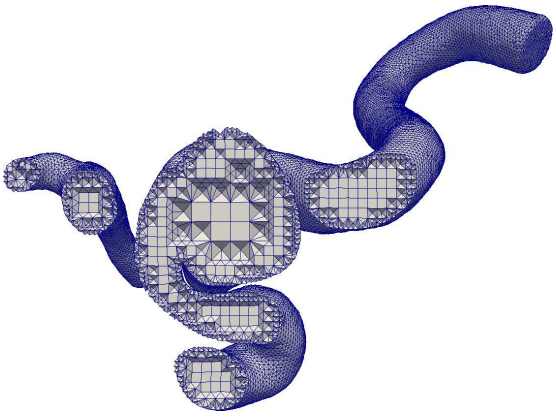
corresponding tissue boundaries to improve smoothness, while taking into account quality and performance aspects.

The present method offers several advantages compared to previous implementations [65, 119]: (i) it improves the accuracy of the geometric and topologic representation, (ii) it provides material-dependent mesh resolution to reduce the element count, (iii) it controls the element quality during mesh smoothing, (iv) it reduces the subsequent memory and CPU requirements for the solver by introducing mixed elements, (v) it reduces the running time using multi-cores, and (vi) it improves the overall reliability and portability of the code as it builds upon the ITK open-source, cross-platform system. A multi-tissue version of the CBC3D is integrated within an interactive simulator for neurosurgical procedures involving brain Arteriovenous Malformations (AVM) developed in SOFA; a framework for real-time medical simulation [64].

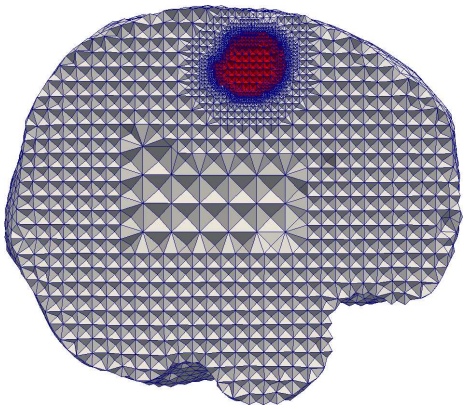
The present method is compared with four other image-to-mesh conversion codes:



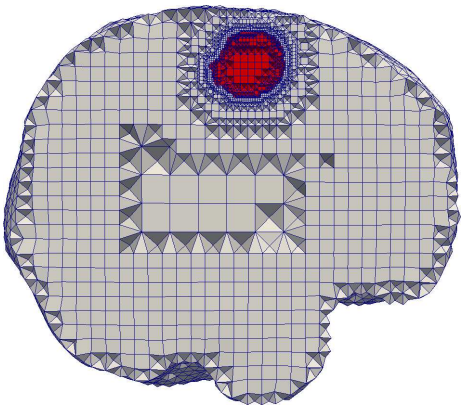
(a) #(pts, tets): (63.6, 272.2) K; α_{min}: 5.07°



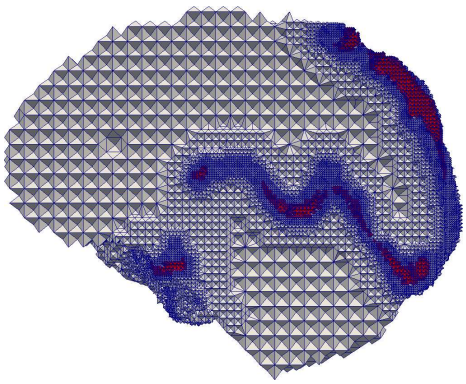
(b) #(pts, tets, pyrs, hexs): (62.1, 246.7, 4.0, 1.4) K; α_{min}: 5.31°; J_{min}: 0.47



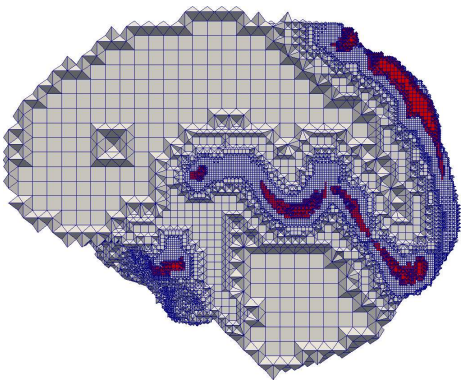
(c) #(pts, tets): (101.4, 578.3) K; α_{min}: 8.89°



(d) #(pts, tets, pyrs, hexs): (78.8, 226.8, 40.2, 22.5) K; α_{min}: 5.19°; J_{min}: 0.55

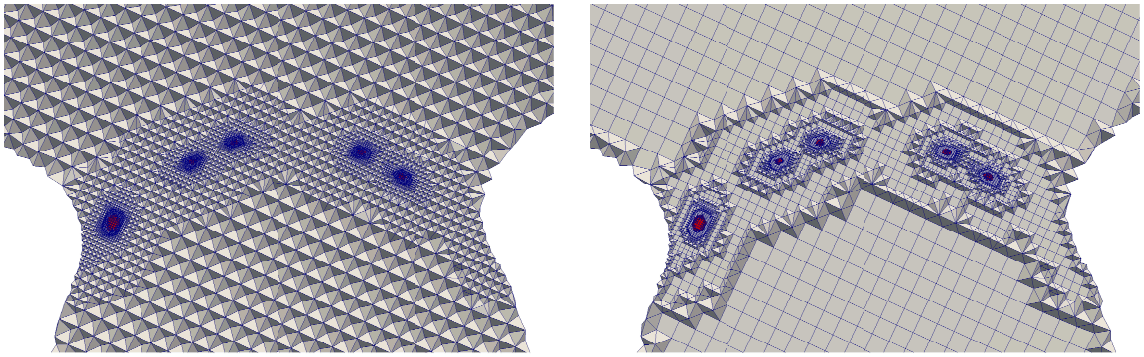


(e) #(pts, tets): (887.0, 5059.9) K; α_{min}: 29.94°



(f) #(pts, tets, pyrs, hexs): (679.4, 1839.1, 364.9, 207.5) K; α_{min}: 29.18°; J_{min}: 0.98

Figure 30: Comparison between adaptive terahedral meshes and their corresponding mixed meshes. All meshes generated with CBC3D. The top, middle, and bottom row correspond to cases 1, 2, and 3, respectively. $\alpha_{min} \in [0, 180]$: minimum dihedral angle (for tetrahedra); $J_{min} \in [0, 1]$: minimum scaled Jacobian (for pyramids and hexahedra).



(a) #(pts, tets): (2.25, 12.98) M; α_{min}: 4.95°

(b) #(pts, tets, pyrs, hexs): (1.57, 3.54, 0.65, 0.67) M; α_{min}: 5.04°; J_{min}: 0.40

Figure 31: Comparison between adaptive terahedral mesh and its corresponding mixed mesh for case 4 (Lumen-LVIS Stent). The meshes generated with CBC3D. $\alpha_{min} \in [0, 180]$: minimum dihedral angle (for tetrahedra); $J_{min} \in [0, 1]$: minimum scaled Jacobian (for pyramids and hexahedra).

CGAL's 3D Mesh Engine (v4.5.2), CLEAVER [26] (v1.5.4), Lattice-Derefinement (LD) [38], and PODM [70]. The evaluation results indicate that the CBC3D meshes: (i) balance the trade-off between mesh conformity and smoothness, (ii) keep the element count reasonably low, and (iii) exhibit a good element quality. In summary, from previous work on lattice-based meshing, it is shown that quality, fidelity, and size criteria are in conflict [38].

Currently, the smoothness of the surface meshes generated from data with small features (e.g. Micro-CT Imaging of stents) is not satisfactory (Figure 22(a)). The smoothness can be improved by incorporating a semi-analytic technique and by using a priori knowledge of the radii of the stent wires. Also, the mechanical properties (i.e, Young's modulus, Poisson ratio) of the materials adjacent on the mesh interfaces can be appropriately adjusted to improve smoothness. The accuracy of CFD simulations to study localized alterations in hemodynamics due to different stent devices may be improved further with boundary layer meshes [61]. Finally, a Structured Adaptive Mesh Refinement (SAMR) scheme [9] can improve the performance and the gradation of the CBC3D method. SAMR, initially discretizes the image domain with a uniform mesh, and then it generates finer sub-meshes (components) near the areas where the physics are "changing". SAMR uses an independent PDE solver in each of the mesh components and accesses the validity of the numerical results. If the numerical solution satisfies the analysis requirements, then the refinement stops; otherwise, SAMR discretizes the mesh with a finer resolution until it

obtains an acceptable solution. Each mesh component has its own solution vector, which is computed independently from the solution of the other mesh components. This is important for the parallelization of the SAMR, where each mesh component is associated to a single core/node, and the under-utilized cores/nodes (e.g. those that compute the PDE solution in smaller number of cells or mesh points) can request automatically additional work from the rest of the cores/nodes.

CHAPTER 3

ADAPTIVE PHYSICS-BASED NON-RIGID REGISTRATION FOR MODELING BRAIN DEFORMATION DURING GLIOMA RESECTION USING PREOPERATIVE AND INTRAOPERATIVE MRI

3.1 OVERVIEW

In image-guided neurosurgery, co-registered preoperative anatomical, functional, and diffusion tensor imaging can be used to facilitate a maximally safe resection of brain tumors in eloquent areas of the brain. However, because the brain can deform significantly during surgery, particularly in the presence of tumor resection, non-rigid registration of the preoperative image data to the patient is required. Using clinical data, this chapter reports the results of a comparison of the accuracy and performance among four open-source non-rigid registration methods for handling brain deformation in the presence of tumor resection, including a new adaptive method that automatically removes mesh elements in the area of the resected tumor, thereby automatically handling deformation in the presence of resection.

This study focuses on thirty glioma surgeries performed at two different hospitals, many of which involved the resection of significant tumor volumes. An adaptive physics-based registration method (Adaptive-PBNRR) registers preoperative and intraoperative MRI for each patient, and the results are compared with three other readily available registration methods: a rigid registration implemented in 3D Slicer v4.4.0; a B-Spline non-rigid registration implemented in 3D Slicer v4.4.0; and a physics-based non-rigid registration (PBNRR) implemented in ITKv4.7.0, upon which Adaptive-PBNRR was based. Three measures aid in assessing the accuracy of the registration methods: (i) visual assessment, (ii) a Hausdorff Distance-based metric, and (iii) a landmark-based approach using anatomical points identified by a neurosurgeon.

The Adaptive-PBNNR using multi-tissue mesh adaptation improved the accuracy of deformable registration by more than 5 times compared to rigid and traditional physics based non-rigid registration, and by more than 4 times compared to B-Spline interpolation methods that are part of ITK and 3D Slicer. Performance analysis showed that Adaptive-PBNRR could be applied, on average, in less than two minutes, achieving desirable speed for use in a clinical setting.

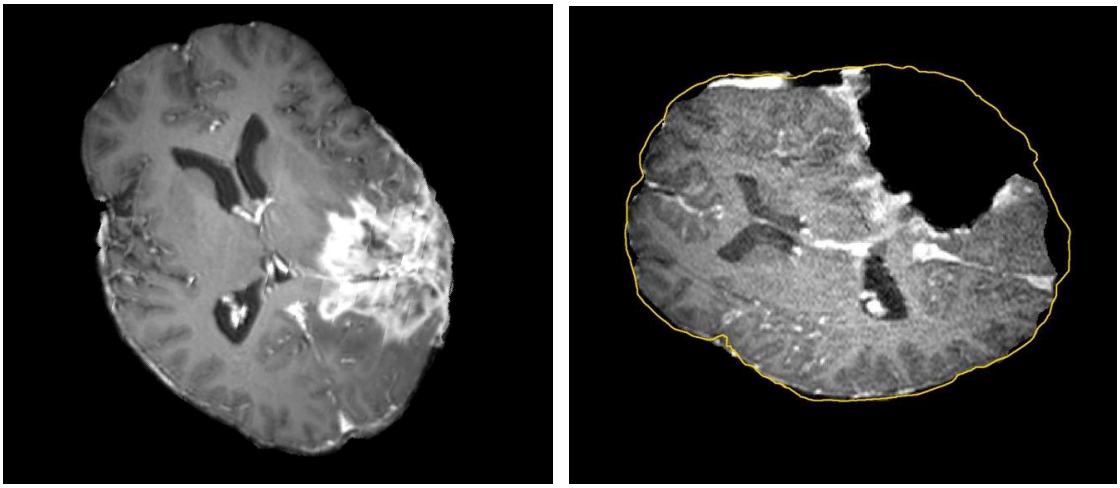
3.2 INTRODUCTION

Malignant gliomas are the most common primary and metastatic brain tumors, accounting for approximately 70% of the 22,500 new cases of primary brain tumors diagnosed annually in adults in the United States [129, 169]. Treatment typically includes surgical removal followed by radiotherapy or chemotherapy. Tumor removal provides a tissue diagnosis, relieves mass effect and intracranial pressure that may be causing pain or other neurological symptoms, and improves prognosis. However, gross total resection is difficult to achieve because of the infiltrative nature of gliomas and because critical functional brain tissue often surrounds tumors. Oncologic outcomes clearly depend on the extent of tumor resection, yet functional preservation is critical for quality of life and survival, so tumor surgery is a delicate balance between removing as much tumor as possible and preserving important functional areas of the brain [103, 136, 172].

During the past two decades, developments in image-guided therapy [85] have allowed surgeons to use preoperative imaging and neuronavigation to facilitate a maximally safe resection of gliomas in eloquent areas of the brain. Preoperative anatomical Magnetic Resonance Imaging (MRI) can be combined with functional MRI (fMRI) to map out areas of the brain near the tumor that are vital to important function such as vision, speech and language, or motor control [92, 80, 93, 111, 153, 197]. Diffusion Tensor Imaging (DTI) can be used to map out white matter tracts that connect to these important regions and run near or through the tumor [17, 16, 82, 63, 99, 150, 186].

Tracking the position of medical tools in patients brain during surgery is possible with neuronavigation using registration of preoperative image data to patient coordinates. The surgeon can then view the location of tools relative to the preoperative anatomical and functional image data, thereby avoiding damage to eloquent areas during tumor resection [4, 45, 57, 113, 122, 139, 151, 148, 32]. Commercial neuronavigation systems (e.g., Stealth by Medtronic and VectorVision by BrainLAB) generally use a rigid transformation to map preoperative image data to patient coordinates. Rigid registration is sufficient when

mapping between rigid objects (e.g., between the skull in the preoperative image data and the patient's skull in the operating room). During surgery, however, the brain deforms due to several factors such as cerebrospinal fluid leakage, intra-cranial pressure, gravity, the administration of osmotic diuretics, and the procedure itself (e.g., tumor retraction and resection) [55, 134, 149]. A rigid transformation will not accurately map preoperative image data to the patients brain during surgery, particularly as the resection proceeds, with the greatest uncertainty at the most critical portions of the surgery (Figure 32).



(a) (b)

Figure 32: Discrepancies between preoperative and intraoperative MR Imaging before and during neurosurgery. (a): preoperative MRI; (b): intraoperative MRI acquired after a part of the tumor is removed. The yellow outline indicates the preoperative brain outline after a rigid rotation. The large dark cavity is the tumor resection.

The adoption of intraoperative MRI (iMRI) has provided a means for monitoring brain deformation during surgery. Over the past decade, the number of hospitals offering iMRI has grown from a handful of research centers to a couple of hundreds clinical sites across the world [19]. Currently, access to iMRI remains limited by high costs, siting challenges, personnel requirements, and disruption of the operative workflow. Several researchers are investigating methods that combine other imaging modalities, such as 3D ultrasound [142, 115, 130, 96] and surface imaging combined with deformation modeling [7, 138, 145], to compensate for brain deformation. However, iMRI remains the gold standard for measuring intraoperative brain deformation and for monitoring tumor resection.

Given an intraoperative anatomical MRI image registered to patient coordinates, preoperative fMRI and DTI image data can be mapped to the intraoperative image and then to

patient coordinates to provide updated guidance for the surgeon. This mapping is usually performed by first registering the fMRI and DTI images to the preoperative anatomical MRI using a rigid transformation, and then registering the preoperative anatomical MRI to the intraoperative MRI image, which is pre-registered to patient coordinates. Because of brain deformation during surgery, registering from preoperative to intraoperative MRI requires a non-rigid registration.

There are several approaches for estimating the non-rigid registration between two or more images, as outlined in prior research [182, 84, 48, 89, 116]. Ranging from control-point registration with spline interpolation to mass-spring models using displacements of anatomical landmarks as force vectors, to physics-based finite element models, many techniques have been applied to non-rigid registration between brain data sets, both for brain mapping [108, 191, 181] and for modeling brain shift [4, 7, 45, 67, 68, 88, 90, 184, 122, 133, 138, 139, 145, 199]. However, most of these methods were not designed to model tissue retraction or resection.

The aim of this study is to evaluate the accuracy and performance of a recently introduced Adaptive-PBNRR for modeling brain deformation in the presence of tumor resection. Appendix A describes a customized Adaptive-PBNRR method to quantify brain shift during Deep Brain Stimulation (DBS) surgery using preoperative and intraoperative CT imaging. The results of a study on 30 glioma cases from two different hospitals are presented, many of which involved the resection of significant tumor volumes. An Adaptive-PBNRR is used to register preoperative and intraoperative MRI for each patient and the results are compared with three methods that are readily available to researchers and clinicians: a rigid registration implemented in 3D Slicer v4.4.0 [98]; a B-Spline non-rigid registration implemented in 3D Slicer v4.4.0 [98]; and a physics-based non-rigid registration (PBNRR) implemented in ITKv4.7.0 [120] upon which Adaptive-PBNRR was based. The accuracy of the registration methods is assessed using three metrics: (i) visual assessment, (ii) a Hausdorff Distance-based metric, and (iii) a landmark-based approach using anatomical points identified by a neurosurgeon. Acceptance in clinical practice requires that non-rigid registration be completed in the time constraints imposed by neurosurgery (e.g., 2-3 minutes) and without the cost of high-performance computing clusters. Thus, we compare processing speeds on a readily-available 12-core desktop system.

3.3 PATIENT POPULATION AND IMAGING PROTOCOLS

This study includes thirty patients from two Hospitals:

1. **Brigham and Women's Hospital (BWH)**. Ten patients (six male, four female) with an age range of 28-62 years and a mean age of 45.2 years, underwent surgery for supratentorial gliomas between April 2005 and January 2006 in Brigham and Women's Magnetic Resonance Therapy (MRT) facility. In some cases, the lesions were in or adjacent to eloquent brain areas, including the precentral gyrus and corticospinal tract for motor function, as well as Broca's and Wernicke's areas for language function.

   (a) **Pre-operative imaging**. The patients underwent the following imaging protocol on a General Electric (Milwaukee,WI) 3T Signa scanner:

      i. Whole brain sagittal 3D spoiled-gradient-echo (SPGR) imaging (slice thickness, 1.3 mm; time of repetition (TR), 6 ms; time of echo (TE), 35 ms; flip angle (FA), 75°; field of view (FOV), 24 cm; matrix size, 256×256).

      ii. Axial T2-weighted fast-spin-echo (FSE) imaging (slice thickness, 5 mm; TE, 100 ms; TR, 3000 ms; FOV, 22 cm; matrix size, 512×512).

   (b) **Intra-operative imaging**. The patients underwent the following imaging protocol on a 0.5T iMR imaging unit (SignaSP; GE Medical Systems, Milwaukee, WI):

      i. Transverse, sagittal, and coronal T1-weighted FSE imaging (TR, 700 ms; TE, 29 ms; FOV, 22 cm; matrix size, 256×256; section thickness, 3 mm, intersection gap, 1mm).

      ii. Transverse T2-weighted FSE imaging (TR, 5000 ms; TE, 99 ms; FOV, 22 cm; matrix size, 256×256; section thickness, 3 mm; intersection gap, 1 mm).

      iii. Transverse 3D SPGR imaging (TR, 15.5 ms; TE, 5.2 ms; FA, 45°; FOV, 22 cm; matrix size, 256×256; section thickness, 2.5 mm; intersection gap, 0 mm).

2. **Huashan Hospital (HSH)**. Twenty patients (eleven male, nine female) with an age range of 19-75 years underwent surgery on single, unilateral, and supratentorial primary gliomas from September 2010 to August 2013. The lesions involved the Pyramid Tracts (PTs) or were in cortical regions in the motor or somatosensory areas, cortical regions adjacent to the central gyrus, subcortical regions with an infiltrative progression along the PTs, and/or deep temporal or insular regions in relation to the internal capsule.

(a) **Pre-operative imaging**. Pre-operative brain images were obtained in the diagnostic room of an iMRI-integrated neurosurgical suite (IMRIS, Winnipeg, Manitoba, Canada) using a ceiling-mounted movable 3.0 T MAGNETOM Verio scanner (Siemens AG, Erlangen, Germany) with a 70 cm working aperture:

i. For suspected high-grade glioma, which showed obvious enhancement after contrast, contrast magnetization-prepared rapid gradient echo (MP-RAGE) was used as the anatomic base for the anisotropic color map and the tumor anatomic feature analysis. T1 contrast images were acquired with a 3D MP-RAGE sequence (TR, 1900 ms; TE, 2.93 ms; inversion time, 900 ms; FA, 9°, FOV, 250×250 mm^2; matrix size, 256×256), after intravenous contrast administration (gadolinium diethylenetriamine penta-acetic acid).

ii. For suspected low-grade gliomas, which showed no obvious enhancement after contrast, a fluid-attenuated inversion-recovery (FLAIR) sequence was used as the anatomic base. T1 contrast images were acquired with FLAIR sequence, with an axial turbo spin echo pulse sequence(TR, 7600 ms; TE, 96 ms; inversion time, 900 ms; FA, 9°; slices, 60; slice thickness, 2 mm; matrix size, 256×180; field of view, 240×240 mm^2).

(b) **Intra-operative imaging**. The same scanner as with preoperative imaging was used:

i. T1 contrast images were acquired with a 3D MP-RAGE sequence (TR, 1900 ms; TE, 2.93 ms; FA, 9°, FOV, 250×250 mm^2; matrix size, 256×215; slice thickness, 1 mm).

ii. T1 contrast images were acquired with FLAIR sequence (TR, 9000 ms; TE, 96 ms; FA, 150°; slice thickness, 2 mm; FOV, 250×250 mm^2; matrix size, 256×160).

A neurosurgeon estimated the volume of resected tumor for each patient by performing a volumetric analysis on the preoperative and intraoperative MRI. Based on this volumetric analysis, the data were categorized as: (i) brain shift (with no resection), (ii) partial resection, (iii) total resection, and (iv) supra total resection. Table 6 summarizes the clinical data. The data collections were carried out with Institutional Review Board (IRB) approval from both hospitals.

Table 6: Clinical MRI data. BWH: Brigham and Women's Hospital; HSH: Huashan Hospital; BS: brain shift; PR: Partial Resection; TR: Total Resection; STR: Supra Total Resection.

Case	Hospital	Genre	Tumor location	Type	Image Size		Image Spacing (mm)	
					Pre-operative	Intra-operative	Pre-operative	Intra-operative
1	BWH	M	Left Perisylvian	BS	$256 \times 256 \times 124$	$256 \times 256 \times 60$	$0.937 \times 0.937 \times 1.30$	$0.859 \times 0.859 \times 2.50$
2	BWH	F	Right Occipital	BS	$256 \times 256 \times 124$	$256 \times 256 \times 60$	$0.937 \times 0.937 \times 1.30$	$0.859 \times 0.859 \times 2.50$
3	BWH	M	Right Frontal	BS	$256 \times 256 \times 124$	$256 \times 256 \times 60$	$0.937 \times 0.937 \times 1.30$	$0.859 \times 0.859 \times 2.50$
4	BWH	F	Left Posterior Temporal	BS	$256 \times 256 \times 124$	$286 \times 286 \times 90$	$0.937 \times 0.937 \times 1.30$	$0.859 \times 0.859 \times 2.50$
5	BWH	M	Left Frontal	BS	$512 \times 512 \times 176$	$256 \times 256 \times 60$	$0.500 \times 0.500 \times 1.00$	$0.859 \times 0.859 \times 2.50$
6	BWH	M	Right Frontal	BS	$256 \times 256 \times 124$	$256 \times 256 \times 60$	$0.937 \times 0.937 \times 1.30$	$0.859 \times 0.859 \times 2.50$
7	BWH	M	Right Occipital	BS	$512 \times 512 \times 176$	$256 \times 256 \times 60$	$0.500 \times 0.500 \times 1.00$	$0.859 \times 0.859 \times 2.50$
8	BWH	F	Left Frontal	PR	$512 \times 512 \times 176$	$256 \times 256 \times 60$	$0.500 \times 0.500 \times 1.00$	$0.859 \times 0.859 \times 2.50$
9	HSH	M	Left Frontal	PR	$448 \times 512 \times 176$	$448 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
10	HSH	M	Left Parietal	PR	$448 \times 512 \times 176$	$512 \times 448 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
11	HSH	M	Right Frontal	PR	$448 \times 512 \times 80$	$512 \times 456 \times 66$	$0.468 \times 0.468 \times 2.00$	$0.468 \times 0.468 \times 2.00$
12	HSH	M	Left Parietal Occipital (deep)	PR	$448 \times 512 \times 176$	$512 \times 448 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
13	BWH	M	Fronto Temporal	TR	$286 \times 286 \times 90$	$286 \times 286 \times 90$	$0.859 \times 0.859 \times 2.50$	$0.859 \times 0.859 \times 2.50$
14	BWH	F	Right Frontal	TR	$256 \times 256 \times 124$	$256 \times 256 \times 60$	$0.937 \times 0.937 \times 1.30$	$0.859 \times 0.859 \times 2.50$
15	HSH	M	Right Temporal	TR	$512 \times 448 \times 176$	$512 \times 448 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
16	HSH	F	Left Posterior temporal	TR	$484 \times 484 \times 58$	$484 \times 484 \times 58$	$0.496 \times 0.496 \times 1.62$	$0.496 \times 0.496 \times 1.62$
17	HSH	F	Left Frontal	TR	$448 \times 512 \times 176$	$448 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
18	HSH	F	Left Frontal	TR	$448 \times 512 \times 176$	$448 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
19	HSH	F	Left Frontal	TR	$448 \times 512 \times 160$	$448 \times 512 \times 160$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
20	HSH	M	Right Frontal	TR	$448 \times 512 \times 88$	$456 \times 512 \times 66$	$0.468 \times 0.468 \times 2.00$	$0.468 \times 0.468 \times 2.00$
21	HSH	M	Left Frontal	TR	$384 \times 512 \times 176$	$512 \times 384 \times 144$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
22	HSH	F	Left Frontal	TR	$448 \times 512 \times 176$	$448 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
23	HSH	F	Left Frontal	TR	$384 \times 512 \times 176$	$384 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
24	HSH	M	Left Occipital	TR	$448 \times 512 \times 176$	$384 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
25	HSH	M	Right Frontal Lobe (deep)	TR	$448 \times 512 \times 176$	$384 \times 512 \times 144$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
26	HSH	M	Right Frontal	STR	$448 \times 512 \times 144$	$448 \times 512 \times 144$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
27	HSH	F	Left Frontal	STR	$384 \times 512 \times 176$	$448 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$
28	HSH	F	Right Frontal	STR	$512 \times 456 \times 66$	$456 \times 512 \times 66$	$0.468 \times 0.468 \times 2.00$	$0.468 \times 0.468 \times 2.00$
29	HSH	F	Right Parietal	STR	$512 \times 456 \times 66$	$512 \times 456 \times 68$	$0.468 \times 0.468 \times 2.00$	$0.468 \times 0.468 \times 2.00$
30	HSH	M	Right Temporal Insular (deep)	STR	$448 \times 512 \times 176$	$448 \times 512 \times 176$	$0.488 \times 0.488 \times 1.00$	$0.488 \times 0.488 \times 1.00$

3.4 SEGMENTATION

Non-rigid registration is performed using a patient-specific brain model derived by segmenting the preoperative anatomical image into brain, tumor, and non-brain regions. Segmentation performance is not critical because preoperative imaging is typically performed a couple of days before surgery. Segmentation was performed with a combination of manual and automatic tools. First, the skull and outer tissues were removed using the open-source Brain Extraction Tool [181]. Further segmentation of the brain surface was performed using a combination of automatic operators implemented in 3D Slicer software (i.e., region growing and level-set filters) [3] and a slice-by-slice manual segmentation correction. While future work will include an evaluation of how segmentation accuracy affects registration accuracy, such analysis is beyond the scope of this thesis.

3.5 MESH GENERATION

The segmentation is used to generate a patient-specific finite element mesh for physics-based non-rigid registration methods. The quality of the tetrahedral mesh influences the numerical accuracy of the solution and the correctness of the estimated transformation. The higher the quality of the elements (i.e., the larger the minimum dihedral angle), the better the convergence of the linear solver. A parallel Delaunay meshing method is employed to tessellate the segmented brain with high quality tetrahedral elements and to model the brain surface with geometric and topological guarantees [70]. Both single-tissue (i.e., brain parenchyma) and multi-tissue (i.e., brain parenchyma and tumor) meshes are generated. Figure 33 depicts one of the multi-tissue meshes. Parameter δ (Table 11) determines the size of the mesh, where a smaller $\delta > 0$ generates a larger mesh.

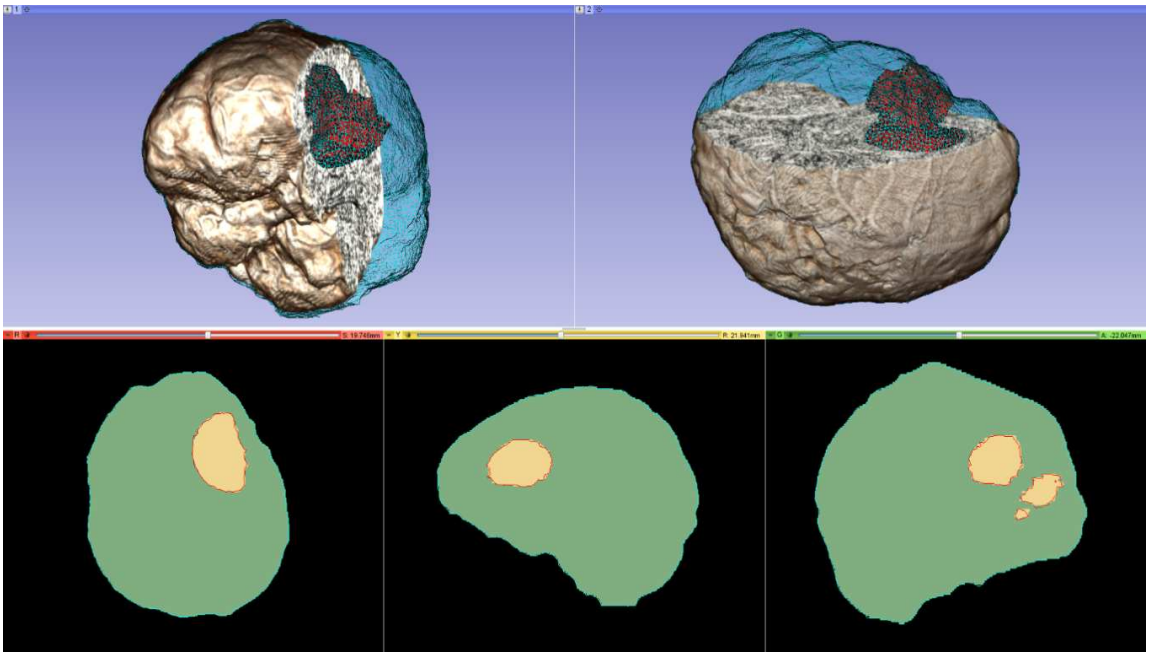


Figure 33: A multi-tissue (brain parenchyma, tumor) finite element mesh used for non-rigid registration (number of tetrahedral elements: 160179; minimum dihedral angle: $4.41°$). Top row: the mesh superimposed on a volume rendering of the MRI data. Cyan and red represent the brain parenchyma and tumor meshes, respectively. Bottom row: mesh fidelity illustrated on an axial, sagittal, and coronal slice. Each slice depicts a 2D cross-section of the mesh surface (cyan and red lines) and the segmented volume (green and yellow regions). The closer the mesh surface is to the segmented boundaries, the higher the mesh fidelity.

3.6 RIGID REGISTRATION

Patients in this study first underwent an intraoperative scan after their head was positioned and fixed for the craniotomy but before the skull was opened. Assuming minimal brain shift at this point, an initial rigid registration was performed to estimate a rigid transformation from the preoperative to intraoperative image data. This rigid transformation was used to initialize non-rigid registration methods. The rigid registration was performed using the BRAINSFit module integrated in 3D Slicer v4.4.0 [98]. BRAINSFit's rigid registration relies on histogram bins and spatial samples to estimate a Mattes Mutual Information cost metric (Table 7). The larger the number of samples, the slower and more precise the fit. The default values for the number of histogram levels and sampling percentage is 50 and 0.2%, respectively. 100 histogram levels and a 5% sampling percentage were selected to achieve higher accuracy (Table 7). Default values were used for the rest of the parameters.

Table 7: Parameters used in this study for rigid registration (RR) and B-Spline non-rigid registration methods implemented in 3D Slicer. MMI: Mattes Mutual Information; VR3DT: Versor Rigid 3D Transform; LBFGSB: Limited memory Broyden Fletcher Goldfarb Shannon minimization with Simple Bounds.

Parameter	Value		Description
	RR	BSPLINE	
Cost metric	MMI		Mattes Mutual Information
Interpolation mode	Linear		-
Sampling percentage	5%		Percentage of image voxels sampled for MMI
Histogram bins	100		Number of histogram levels
Optimizer type	VR3DT	LBFGSB	-
Max number of iterations	1500		Maximum number of optimizing iterations
Grid size	-	$15 \times 15 \times 15$	Number of grid subdivisions
Min step length	10^{-3}		Min threshold step for optimizer
Projected gradient tolerance	-	10^{-5}	Used by LBFGSB

3.7 NON-RIGID REGISTRATION

As surgery proceeds, the initial rigid preoperative to intraoperative registration becomes increasingly less valid. To make the best use of preoperative image data for surgical guidance (including the use of fMRI and DTI to inform the surgeon of critical structures near the tumor), the preoperative image data must be transformed to the patient's coordinate system using non-rigid registration.

Recent efforts have aimed to model intraoperative brain deformation during tissue retraction and tumor resection. [137] introduced a method for modeling retraction and resection using a multi-step procedure, which allows arbitrary placement and movement of a retractor and removal of tissue. Tissue resection is modeled by manually deleting model elements identified as tumor in the preoperative image. [164] proposed a registration framework based on the bijective Demons algorithm which can handle retraction and resection. Retraction is detected at areas of the deformation field with high internal strain and the estimated retraction boundary is integrated as a diffusion boundary in an anisotropic smoother. Resection is detected by a level set method evolving in the space where image intensities disagree. [67] tracked brain deformation due to tumor resection over multiple intraoperative MRI acquisitions. After each scan, the brain surface is segmented, and a surface-matching algorithm is used to drive the deformation of a finite element model of the brain. [194] modeled retraction by segmenting the brain surface from two sequential intraoperative MRI image volumes and identifying landmarks on these surfaces. Displacements of the landmarks between the two surfaces are used to drive deformation using a finite element modeling technique that allows discontinuities at the resection boundary. We recently introduced [58, 57, 60] an adaptive form of a Physics-Based Non-Rigid Registration (PBNRR) method originally described by [45] and updated and implemented in ITK by Liu [120, 122]. This Adaptive-PBNRR was specifically developed to model tumor resection in intraoperative images without manual intervention. Unlike other methods [67, 194] it uses image-based registration and thereby does not require segmentation of the intraoperative MRI image, which is time-consuming and may require manual intervention. The algorithm is parallelized to ensure that it is fast enough to be useful in a clinical setting.

As a standard for comparison, both the rigid and the non-rigid registration methods were implemented in the open-source BRAINSFit module of a 3D Slicer. The non-rigid registration method is based on a B-spline interpolation scheme, which uses a 3-dimensional cubic control grid to optimize the registration [98]. Table 7 lists the parameters for the B-Spline deformable registration method. To facilitate performance comparisons, all three non-rigid registration methods are parallelized for shared memory multiprocessor architectures.

3.7.1 PHYSICS-BASED NON-RIGID REGISTRATION (PBNRR)

PBNRR has been shown to accurately capture brain shift (i.e., volumetric deformations of the brain) during image-guided neurosurgery [45]. An open-source implementation of

this method, which exploits the GPU and multi-core accelerators to improve performance is presented in [120]. PBNRR uses the finite element method (FEM) to model deformations and estimate a sparse displacement vector associated with selected features located in the cranial cavity of the preoperative image. The sparse vector is used (as boundary conditions) to drive the deformation of a patient-specific, single-tissue (i.e., brain parenchyma), finite element mesh. PBNRR includes three modules: Feature Selection, Block Matching, and the FEM Solver.

Feature Selection

PBNRR identifies image features algorithmically by analyzing voxel intensity variation across the intra-cranial cavity. For each feature candidate, it computes the variance within a block of size B_s (Table 11). It then selects F_s features with the highest variance. Experimental evaluation has shown that when $B_s = 3$, or $B_s = 5$ and $F_s \in [0.03, 0.10]$, a sufficient number of image blocks ($> 3 \times 10^5$) can be selected. The method also uses a connectivity pattern (Table 11) to avoid selecting blocks that are too close to each other, thereby influencing the distribution of selected blocks in the image. Three simplex-patterns are available: "vertex" (i.e., zero-order simplex implies 26 connectivity), "edge" (i.e., first-order simplex implies 18 connectivity), and "face" (i.e., second-order simplex implies 6 connectivity). The higher the order of the simplex-pattern the higher the density of the selected blocks. Figure 34 depicts feature selection results using the three patterns. Since the "face" pattern results in a higher density of blocks near the boundaries/interfaces of anatomical structures, features are expected to be most persistent between preoperative and intraoperative image acquisitions, which is suitable for this application.

Block Matching

Given a block in the preoperative image and a block matching window W of size W_s in the intraoperative image, this module searches for a block in W that maximizes the similarity between the block in the preoperative image and the block in W. Displacements between corresponding blocks in the preoperative and intraoperative images are used to drive the FEM solver. W_s was chosen empirically after visual inspection of the preoperative and intraoperative images. Larger windows were chosen for complete and extensive tumor resections, and smaller windows were chosen for brain shift and partial resections (Table 11). Note that W_s may be different in the axial, coronal, and sagittal directions due to anisotropic image data.

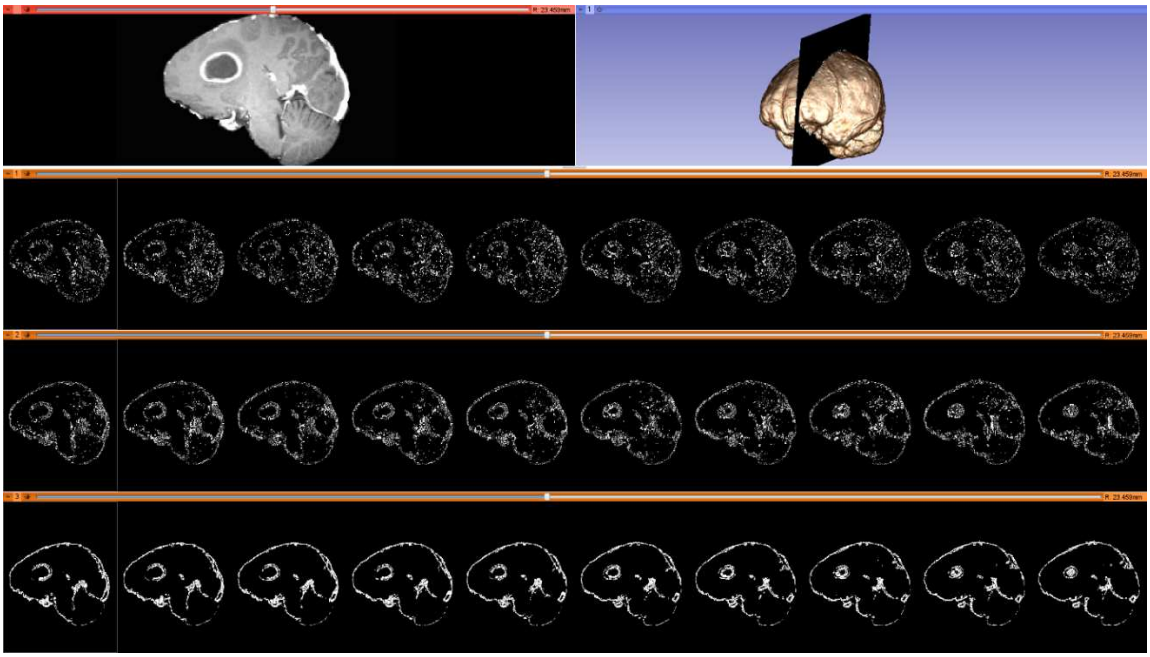


Figure 34: Selected blocks from an MRI volume using various connectivity patterns. Blocks are depicted on ten consecutive sagittal slices. From top to bottom row: sagittal slice (left) and volumetric MRI rendering (right); selected blocks with a "vertex" pattern; selected blocks with an "edge" pattern; selected blocks with a "face" pattern. Number of selected blocks for all patterns: 322060.

FEM Solver

This module assembles an $N \times N$ global stiffness matrix $K_g = K_m + K_b$ for the biomechanical brain model, where $N = 3 \times n$ is the number of degrees of freedom in the model, n is the number of mesh vertices, and K_m, K_b are the stiffness matrices of the mesh and the F_s selected blocks, respectively. The stiffness matrix depends on the geometry/quality of the mesh, the interpolation polynomial-order (e.g., linear), and the mechanical properties (E_b, v_b) of the underlying materials (Table 11). Matrix K_m is assembled from the individual stiffness matrices of mesh elements [15]. Matrix K_b represents the stiffness of the blocks. After associating each block with an element, a 3×3 stiffness tensor is computed by interpolating the element stiffness at the block position [45]. Matrix K_b is assembled from the individual 3×3 block tensors.

The FEM solver constrains the mesh model by applying displacements determined

during block matching, represented by external force vector F, to the associated mesh elements. The displacements U of unconstrained mesh vertices are then estimated by solving an $N \times N$ linear system of equations $F = K_g \times U$. The FEM solver uses ITPACK structures for matrix representation and a Jacobi Conjugate Gradient (JCG) iterative solver [107] to compute U. Mesh deformations are estimated using an approximation of an interpolation based formulation which rejects feature outliers (i.e., blocks with the largest error between U and the block matching displacements). In each outlier rejection step, $(N_b \times F_r)/N_{rej}$ outliers are removed, where N_b is the number of selected blocks, F_r is the fraction of the rejected blocks, and N_{rej} is the number of outlier rejection iterations (Table 11). Previous experiments have shown that $F_r = 25\%$ is sufficient to reject all significant outliers without rejecting relevant matches. Deformation at each image voxel can be estimated by interpolating the final solution U across mesh elements.

3.7.2 ADAPTIVE PHYSICS-BASED NON-RIGID REGISTRATION (ADAPTIVE-PBNRR)

Adaptive-PBNRR [57] augments PBNRR [120] to accommodate soft-tissue deformation caused by tumor resection. Figure 35 illustrates the adaptive framework. This iterative method adaptively modifies a heterogeneous finite element model to optimize non-rigid registration in the presence of tissue resection. Using the segmented tumor and the registration error at each iteration, Adaptive-PBNRR gradually excludes the resection volume from the model. During each iteration, registration is performed, the registration error is estimated, the mesh is deformed to a predicted resection volume, and the brain model (minus the predicted resection volume) is re-tessellated. Re-tessellation is required to ensure high quality mesh elements. The following paragraphs describe the major improvements of Adaptive-PBNRR over PBNRR.

Adaptivity

An adaptive, iterative process allows Adaptive-PBNRR to gradually change the preoperative model geometry to accommodate resection. Figure 36 illustrates five iterations of Adaptive-PBNRR on a brain with a significant resection volume. The mesh in the beginning of iteration i is denoted by M_i. Initially, a multi-material mesh is generated from the input segmentation. After the completion of iteration i, the displacements are computed on the vertices of M_i, which yields the deformed mesh M_i'. M_i' is not used for the next

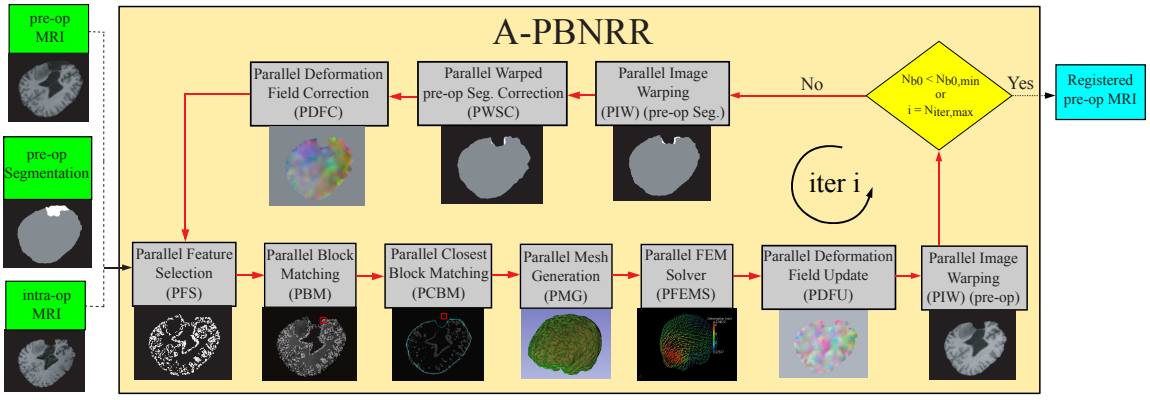


Figure 35: The Adaptive-PBNRR framework. Green, cyan, and gray represent the input, the output, and the modules of the framework, respectively. The modules are built on the ITK open-source system and are parallelized with the POSIX thread library for shared memory multi-core machines. The red arrows show the execution order of the different modules in the loop. The loop breaks when the number of blocks with zero displacements (N_{b_0}) is less than a minimum $(N_{b_0,min})$, or when the maximum number of iterations $(N_{iter,max})$ has been reached.

loop because it contains distorted elements. Instead, a corrected warped image segmentation is used in the next iteration as the input for the mesher. The mesher generates in parallel a new global tesselation M_{i+1} with good shaped elements. Parameter δ (Table 11) determines the size of the mesh ($\delta > 0$). The smaller the δ, the larger the mesh. Table 8 illustrates how the remeshing eliminates the elements with small and large dihedral angles for an example of four adaptive iterations. "Before" corresponds to the deformed mesh at $i-1$, and the "After" to the new generated mesh at i. The #Tets refers to the number of tetrahedral elements in the new generated mesh at i. Generally, the higher the quality of the elements, the better the conditioning of the stiffness matrices and thus the convergence of the linear solver.

Heterogeneity

Whereas PBNRR can only be applied to a homogenous (single-tissue) brain model, Adaptive-PBNRR can accommodate a heterogeneous (multi-tissue) model. This study involved two-tissue models (brain parenchyma and tumor), but the method can accommodate any number of tissues. Figure 33 depicts a heterogeneous brain model, and Table 11

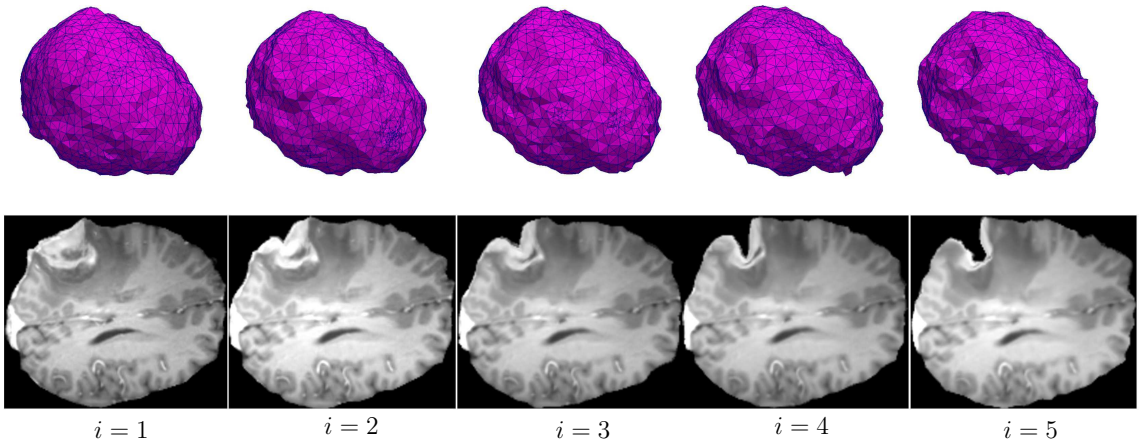


Figure 36: Non-rigid registration using 5 iterations of Adaptive-PBNRR reflects the changes in brain morphology caused by tumor resection. Each column depicts a brain mesh model (top row) and an axial slice of the preoperative image after the non-rigid registration (bottom row) at iteration i.

Table 8: The minimum and maximum dihedral angles α, before and after the Parallel Mesh Generation ($\delta = 5$).

Iteration i	#Tets	α_{min}		α_{max}	
		Before	After	Before	After
1	14278	-	5.00°	-	169.68°
2	13482	0.34°	5.24°	179.28°	169.92°
3	13497	0.14°	4.91°	179.79°	169.71°
4	12957	0.10°	5.01°	179.80°	171.32°

lists the mechanical tissue properties of this study.

Enhanced Block Matching (PBM+PCBM)

In tumor resection, the problem of blocks with a missing correspondence may compromise the accuracy of the non-rigid registration. A large window size cannot solve the problem because: (i) it is opposed to the assumption that a complex non-rigid transformation can be approximated by point-wise translations of small image regions, (ii) it may lead to unrealistic matches that deteriorate the quality of the transformation, and (iii) it imposes performance overheads due to an exhaustive search on a large number of blocks.

The Adaptive-PBNRR introduces a heuristic two-step block matching algorithm to eliminate the blocks with missing correspondence. First, a Parallel Block Matching (PBM) algorithm maximizes the similarity between blocks in the preoperative image and blocks in W [120]. Next, a Parallel Closest Block Matching (PCBM) algorithm pairs the blocks with a missing correspondence with their closest point on the surface $\mathcal{S}$ of the brain in the intraoperative image (Figure 37). For each block i with displacement $\|d_i\| = 0$ its minimum Euclidean distance to $\mathcal{S}$ is computed; that is $\|d_{i,\mathcal{S}}\|$ where $\| * \|$ is the Euclidian norm. The maximum displacement due to window W is:

$$\|d_{max}\| = \sqrt{\left(s_x \left\lfloor \frac{W_{s,x}}{2} \right\rfloor\right)^2 + \left(s_y \left\lfloor \frac{W_{s,y}}{2} \right\rfloor\right)^2 + \left(s_z \left\lfloor \frac{W_{s,z}}{2} \right\rfloor\right)^2} \tag{3}$$

where s is the image spacing (Table 6) and W_s the window size (Table 11). The block displacement vector d_i is scaled:

$$d_i \leq \frac{\|d_{max}\|}{\|d_{i,\mathcal{S}}\|} \cdot d_{i,\mathcal{S}} \tag{4}$$

where $\|d_{max}\|/\|d_{i,\mathcal{S}}\| < 1$ and $\|d_{i,\mathcal{S}}\| \neq 0$. In practice, the upper bound in (4) is chosen. Figure 37 depicts a block displacement vector before and after scaling.

Parameter $N_{b_0,min}$ (Figure 35, Table 11) controls the minimum allowed number of blocks with a missing correspondence. Based on the experimental evaluation in this study, $N_{b_0,min} = 1\% \cdot N_b$ results to high registration accuracy within a few adaptive iterations (N_b is the number of selected blocks from feature selection). PCBM computes the block displacements in parallel after distributing $\lfloor N_{b_0}/k \rfloor$ blocks among $k-1$ threads and $N_{b_0} - (k-1) \cdot \lfloor N_{b_0}/k \rfloor$ blocks to the last thread.

Similarity measures in registration include Mean Square Difference of intensity (MSD), Mutual Information (MI), and Cross Correlation (CC) [27]. Adaptive-PBNRR enhances the PBM module [120] with a Normalized Mutual Information ($NMI \in [0, 1]$) metric [183]. A value close to one indicates that two blocks are very similar. A value close to zero indicates that the blocks are very different. NMI is computed by the following equation [183]:

$$NMI(B_{pre}, B_{intra}) = 1 - \frac{H(B_{pre}, B_{intra})}{H(B_{pre}) + H(B_{intra})} \tag{5}$$

where $H(B_{pre})$ and $H(B_{intra})$ are the marginal entropies of a block in the preoperative and the intraoperative image, respectively. $H(B_{pre}, B_{intra})$ is the joint entropy of the two blocks. The maximization of (5) seeks a transformation where joint entropy is minimized

with respect to the marginal entropies. NMI is suitable in multi-modal registration, where the affine hypothesis between the intensity distributions of the two images is hardly valid. This study concerns images of the same modality, thus the NCC metric is used. In future work, the NMI metric will be used for the non-rigid registration of MRI to CT scans.

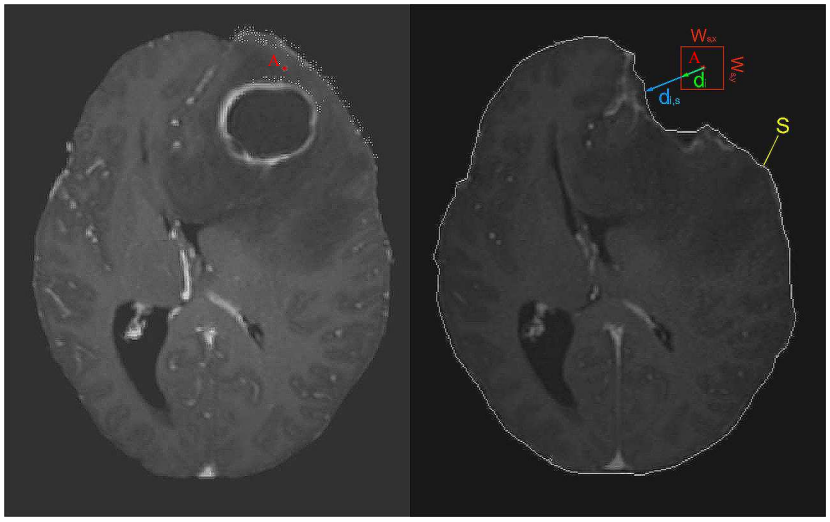


Figure 37: PCBM module on a pair of brain MRIs. Left: preoperative MRI; right: intraoprative MRI. The white points on the top right of the left figure indicate blocks without a correspondence. A is one of those blocks. $\mathcal{S}$ is the set of points extracted from the brain surface in the intraoperative image. $d_{i,\mathcal{S}}$ (blue) is the Euclidian distance vector from the center of the block to its closest point on $\mathcal{S}$. d_i (green) is the scaled Euclidian distance vector.

Higher Parallelization

Adaptive-PBNRR uses a parallel framework that can target shared memory multi-core machines. A previous study [57] showed that Adaptive-PBNRR exploits additional parallelism over PBNRR with corresponding performance improvements so that, even with multiple iterations, Adaptive-PBNRR requires on average less than two minutes to perform non-rigid registration. The following paragraphs present details of the new parallel modules in the non-rigid registration procedure.

Parallel Feature Selection (PFS)

Adaptive-PBNRR identifies the image features in parallel. Initially, the input image is partitioned into k sub-regions (k is the number of threads). Each thread computes a pair

of a variance value and an image index for all the feature blocks inside its sub-region. The computed pairs are sorted in parallel according their variance values and merged into a global vector. The size of the global vector is equal to the total number of image blocks $N_{b,tot}$. Next, $\lfloor 0.5 + N_{b,tot} \cdot F_s \rfloor$ feature blocks are selected from the global vector in parallel.

It should be noted that the parallel algorithm selects feature blocks from the global vector with a different order compared to the sequential algorithm. Also, a prohibited connectivity is used [120] to avoid overlaps between selected blocks (Figure 34). After a block is selected, its neighboring blocks are not candidates for selection. For these reasons, the output blocks of the parallel procedure are in general different from the output blocks of the sequential procedure.

The global vector is first partitioned into sub-vectors to concurrently select features from the global vector using k threads. The number of partitions can be adjusted with parameter $npar$. The larger the $npar$, the smaller the size of each sub-vector and the more similar the selected features of the parallel and the sequential algorithm. However, very large values for $npar$ increase the execution time of the parallel implementation because thread synchronization is more frequent.

For an average 3D brain MRI with $N_{b,tot} \approx 10^6$, the experimental evaluation indicated that a good balance between the speed of the selection and the similarity between the output blocks in the parallel and the sequential implementation can be achieved with $npar \in [100, 200]$. For the experimental evaluation in this study, a value of $npar = 100$ was used.

Parallel FEM Solver (PFEMS)

The FEM Solver estimates the deformation U on the mesh vertices using a hybrid approach: from an approximation to an interpolation-based formulation [45]. In [118] the FEM Solver was parallelized using ParMETIS[1] for a balanced parallel partitioning of the tetrahedral mesh among the processing cores, and PETSc [11] to assemble and solve the linear system of equations. In [120] the FEM Solver was integrated in ITK using ITPACK's matrix structures [107]. The Adaptive-PBNRR:

1. Parallelizes components of the FEM Solver [120].

2. Incorporates the iterative solvers Jacobi Conjugate Gradient (JCG) [107],

[1] http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

LSQR [154], and BICGSTAB [13], and a direct solver based on Lower-Upper decomposition with forward elimination and back substitution [104], to solve the linear system $F = K_g \times U$.

3. Incorporates the isoparametric linear/quadratic, tetrahedron/hexahedron element, and the hexahedron element with Extra Shape Functions (ESF); those elements are widely used in commercial FEM software packages (e.g., Ansys[2] and Abaqus[3]).

Table 9 summarizes the capabilities of the FEM Solver within PBNRR, and Adaptive-PBNRR.

Table 9: Comparison of the features of FEM Solver integrated within PBNRR and Adaptive-PBNRR. JCG: Jacobi Conjugate Gradient [107]; LSQR: Least Squares [154]; BICGSTAB: Biconjugate Gradient Stabilized [13]; LU: Lower-Upper decomposition with forward elimination and back substitution [104]; Tet: Tetrahedron; Hex: Hexahedron; ESF: Extra Shape Functions.

Method	Parallel	Storage			Linear Solver			Element Type					
		ITPACK	VNL	JCG	LSQR	BICGSTAB	LU	Tet		Hex		Hex+ESF	
								4-node	10-node	8-node	20-node	8-node	20-node
PBNRR		✓	✓	✓	✓			✓		✓			
A-PBNRR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

PFEMS includes the initialization, the assembly, the outlier rejection, and the interpolation step. The initialization step computes in parallel an image interpolation grid and the block tensors. The image interpolation grid is useful for the calculation of the output deformation field. The block tensor provides a 3-dimensional measure of the smoothness of the intensity distribution in a block [45]. Each voxel in the interpolation grid is associated with its containing element. A voxel is contained within an element if all areal coordinates of the geometric center of the voxel are less than one. For the parallel computation, $\lfloor Ne/k \rfloor$ elements are distributed among $k-1$ threads and the rest $Ne - (k-1) \cdot \lfloor Ne/k \rfloor$ elements are assigned to the last thread. Then each thread computes the contained voxels within its elements in parallel. The block tensors are computed in the same manner, but instead of mesh elements, N_b blocks are distributed among the threads.

To assemble the stiffness matrix K_b of N_b selected blocks, first $\lfloor N_b/k \rfloor$ blocks are distributed among $k-1$ threads and $N_b - (k-1) \cdot \lfloor N_b/k \rfloor$ blocks are assigned to the last thread. Each thread assembles a stiffness matrix K_{b_i} ($i = 1, 2, \ldots, k$). The matrices K_{b_i} are recursively added using a maximum $\lfloor k/2 \rfloor$ threads. In the first recursion $\lfloor k/2 \rfloor$

[2]http://www.ansys.com/

[3]http://www.3ds.com/products-services/simulia/products/abaqus/

threads are used. The first thread adds K_{b_1} and K_{b_2} and stores the result to $K_{b_{12}}$; the second thread adds K_{b_3} and K_{b_4} and stores the result to $K_{b_{34}}$; ...; thread $\lfloor k/2 \rfloor$ adds $K_{b_{k-1}}$ and K_{b_k} and stores the result to $K_{b_{(k-1)k}}$. In the second recursion, $\lfloor k/4 \rfloor$ threads are used. The first thread adds $K_{b_{12}}$ and $K_{b_{34}}$; the second thread adds $K_{b_{56}}$ and $K_{b_{78}}$; ...; thread $\lfloor k/4 \rfloor$ adds $K_{b_{(k-3)(k-2)}}$ and $K_{b_{(k-1)k}}$. For the last recursion one thread is used. The matrix $K_{b_{12\ldots k/2}}$ is added to the matrix $K_{b_{(k/2+1)(k/2+2)\ldots k}}$ and the result is stored into the matrix K_b. A similar approach is employed to assemble the mesh stiffness matrix K_m, but mesh elements are distributed among the threads instead of blocks. The matrices K_b and K_m are added sequentially to compute the stiffness matrix K_g of the biomechanical brain model.

In each outlier rejection step, $(N_b \cdot F_r)/N_{rej}$ blocks with the highest error are removed, where N_b is the number of selected blocks, F_r is the fraction of the rejected blocks, and N_{rej} is the number of outlier rejection iterations (Table 11). PFEMS computes the outliers with the highest error in parallel, i.e., $N_b - j \cdot (N_b \cdot F_r/N_{appr})$ blocks are distributed among the threads, and each thread calculates the error for its own blocks. The *nth_element* function of the GNU libstdc++ library is used to sort in parallel the outliers based on their error. Because outlier blocks are rejected in each approximation iteration, their contribution should be also removed from matrix K_b. K_b is modified in parallel after distributing $N_b \cdot F_r/N_{appr}$ outliers among k threads. Each thread builds a block outlier matrix K'_{b_i}, $(i = 1, 2 \ldots k)$, and K'_{b_i} is recursively subtracted from K_b. The result is stored within matrix K_b. Finally, the matrices K_b and K_m are added sequentially to compute the updated stiffness matrix K_g of the biomechanical model.

Table 10 compares the accuracy and the speed of the integrated solvers using different fractions F_s. When a sufficient fraction of blocks is selected ($F_s = 5\%$), all solvers converge to the same $\|\mathrm{U}\|$. JCG and BICGSTAB converge rapidly, hence they are more suitable for time critical applications. However, when the fraction is too small ($F_s = 0.1\%$) the JCG iterative solver fails to converge. Some conditions affecting the convergence of JCG may be positive diagonal elements, diagonal dominance, and symmetry of the system [107]. Figure 38 illustrates the assembled stiffness matrices K_m, K_b, and $K_g = K_m + K_b$ for the experiment in Table 10. BICGSTAB is the default solver in Adaptive-PBNRR.

Parallel Deformation Field Update (PDFU)

The deformation field is a transformation that maps each point in the intraoperative image to its corresponding point in the preoperative image. Each voxel in the deformation

Table 10: Accuracy and timing performance of the integrated solvers in A-PBNRR. A sparse 8391×8391 linear system $F = K_g \times U$ was solved. F_s: % selected blocks from the total number of blocks; r: residual norm; $\|\mathrm{U}\|$: ℓ^2 norm of the solution vector; JCG: Jacobi Conjugate Gradient; LSQR: Least Squares; BICGSTAB: Biconjugate Gradient Stabilized; LU: Lower-Upper decomposition with forward elimination and back substitution. The experiments performed on a workstation with Intel Xeon X5690@3.47 GHz CPU and 96 GB of RAM.

F_s	Linear solver	r	$\|\mathrm{U}\|$	Time (sec)
5%	JCG	$4.90 \cdot 10^{-6}$	45.530	0.12
	LSQR	$1.02 \cdot 10^{-12}$	45.530	24.03
	BICGSTAB	$8.49 \cdot 10^{-6}$	45.530	0.12
	LU	n/a	45.530	213.50
0.1%	JCG	$6.95 \cdot 10^{-2}$	101.871	0.18
	LSQR	$4.86 \cdot 10^{-5}$	43.134	46.12
	BICGSTAB	$9.89 \cdot 10^{-6}$	43.108	0.12
	LU	n/a	43.136	207.52

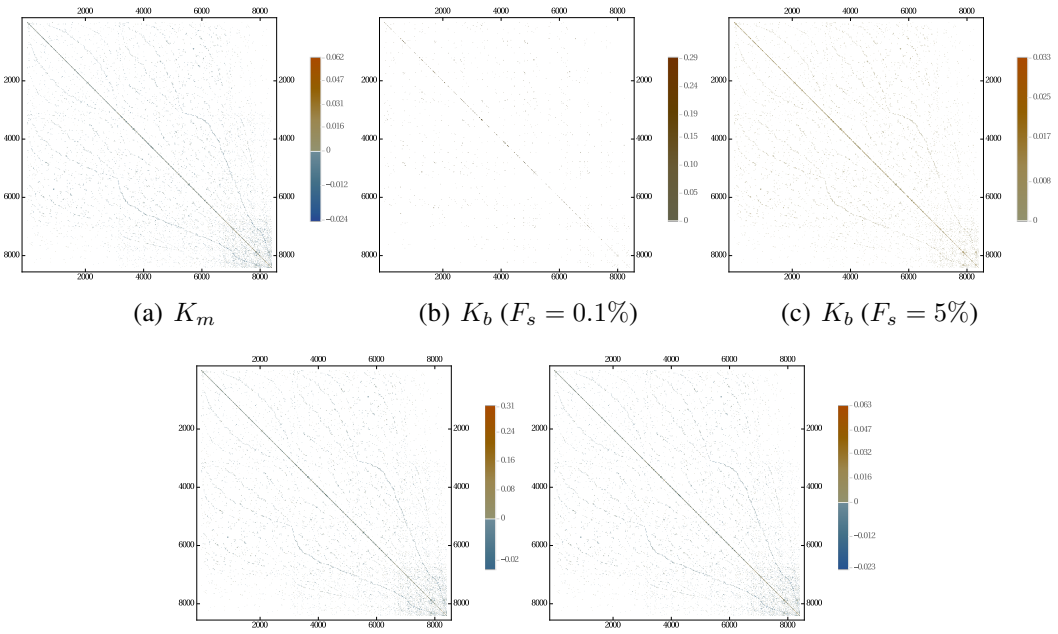
field has a constant value, i.e., a displacement vector. The Adaptive-PBNRR method uses an additive deformation field for image warping. Therefore, only the preoperative image is interpolated, independently of the number of adaptive iterations. The additive field at the adaptive iteration i is computed from the additive field at previous iteration $i-1$ and the output field at iteration i:

$$DF'_i = DF'_{i-1} + DF_i \tag{6}$$

DF'_i, and DF'_{i-1} is the additive image deformation field at iteration i and $i-1$, respectively. DF_i is the output deformation field at iteration $i > 0$. The calculation of DF_i and the addition in (6) are performed in parallel. First, the deformed mesh is decomposed into k sub-meshes, where k is the number of the threads. Next, the displacements at the geometric centers of voxels contained within an element in a sub-mesh are computed. The displacement vectors at images DF'_{i-1} and DF_i are added in parallel after the images are partitioned into k sub-images.

Parallel Image Warping (PIW) - Parallel Warped pre-op Segmentation Correction (PWSC) - Parallel Deformation Field Correction (PDFC)

ITK's multi-threaded filter is employed for image warping. An image is warped by using the additive deformation field in (6) and an ITK interpolator to estimate the image



(a) K_m (b) K_b ($F_s = 0.1\%$) (c) K_b ($F_s = 5\%$)

(d) K_g ($F_s = 0.1\%$) (e) K_g ($F_s = 5\%$)

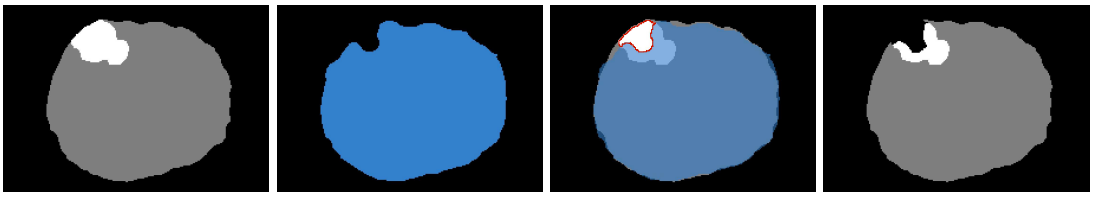
Figure 38: Sparse stiffness matrices of the experiment in Table 10. K_m: mesh stiffness matrix; K_b: blocks stiffness matrix; $K_g = K_m + K_b$: biomechanical stiffness matrix; F_s: % selected blocks from the total number of blocks.

values at the non-integer pixel positions. A linear interpolator and a nearest neighbor interpolator is used to warp the preoperative MRI and the preoperative segmented image, respectively.

The PWSC module accounts for the resected tissue in the segmented image with heuristics. The warped segmented image at iteration i and a mask of the intraoperative image are overlapped, and the portion of the tumor that falls into the mask background is relabeled (Figure 39). To parallelize PWSC module, the segmented image is partitioned into k pieces, and the overlapping/relabelling is performed concurrently with k threads. The PDFC module modifies the additive deformation field DF'_i at (6) to take into account the changes due to relabelling. For this purpose, the displacement vectors of the voxels contained within the relabeled region are initialized to zero. PDFC is performed in parallel after the partition of the image field DF'_i into k pieces.

3.8 RESULTS

An evaluation of four registration methods is performed using imaging data from thirty



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

Figure 39: The correction of a warped segmented image at adaptive iteration i. Gray and white represent the brain parenchyma and the tumor, respectively. Blue signifies the mask of the intraoperative image. (a): warped segmentation, (b): intraoperative mask, (c): (a) and (b) overlapped, (d): warped segmentation after relabelling.

patients who underwent partial, total, and supra total glioma resection. Visual inspection, a Hausdorff Distance-based error metric, and a landmark-based error measured by Dr. Chengjun Yao were utilized to assess accuracy. The four registration methods were:

1. Rigid registration implemented in 3D Slicer v4.4.0 [98].
2. B-Spline non-rigid registration implemented in 3D Slicer v4.4.0 [98].
3. Physics-based non-rigid registration (PBNRR) implemented in ITKv4.7.0 [120].
4. A modification of PBNRR than handles tumor resections (Adaptive-PBNRR) [57].

Tables 7, 11 list the input parameters used for the registration methods

3.8.1 VISUAL ASSESSMENT

In most applications, careful visual inspection remains the primary validation check. Figures 40-41 present the registration results for thirteen representative tumor resection cases (three partial, seven total, and three supra total resections). For each patient, Figures 40-41 show a 2D section from the intraoperative MRI, the corresponding registered preoperative MRI, and the difference between the intraoperative and registered preoperative MRIs. Smaller differences indicate a more precise alignment. As Figures 40-41 illustrate, Adaptive-PBNRR provides the most accurate registration and preserves brain morphology in the presence of resection, specifically near tumor margins. Registration accuracy is not significantly affected by whether the resection was partial, total, or supra total. In contrast, the other registration methods fail to capture the complex soft-tissue deformation near the tumor resection. These results were confirmed by a neurosurgeon who

Table 11: Parameters used for PBNRR and Adaptive-PBNRR. BS: brain shift; PR: Partial Resection; TR: Total Resection; STR: Supra Total Resection.

Parameter	Value	Description
Initialization transform	Rigid	Rigid transformation to initialize the non-rigid registration
Connectivity pattern	"face"	Pattern for the selection of blocks
F_s	5%	% selected blocks from total number of blocks
$B_{s,x} \times B_{s,y} \times B_{s,z}$	$3 \times 3 \times 3$ $7 \times 7 \times 3$ (BS)	Block size (in voxels)
$W_{s,x} \times W_{s,y} \times W_{s,z}$	$9 \times 9 \times 3$ (PR) $13 \times 13 \times 3$ (TR, STR)	Block matching window size (in voxels)
δ	5	Mesh size
E_b	2.1 KPA	Young's modulus for brain parenchyma
E_t	21 KPA	Young's modulus for tumor (A-PBNRR)
v_b	0.45	Poisson ratio for brain parenchyma
v_t	0.45	Poisson ratio for tumor (A-PBNRR)
F_r	25%	% of rejected outlier blocks
N_{rej}	10	Number of outlier rejection steps
$N_{iter,max}$	10	Max number of adaptive iterations (A-PBNRR)
$N_{b0,min}$	$1‰\times$ number of selected blocks	Min number of blocks with zero correspondence (A-PBNRR)

inspected the full registered volumes.

3.8.2 QUANTITATIVE ASSESSMENT USING HAUSDORFF DISTANCE (HD)

An objective and automatic method [76] was employed to quantitatively evaluate the registration accuracy. This method uses Canny edge detection [29] to detect two point sets. The first point set is extracted from the preoperative volume Figure 42(a) and transformed (using the deformation field computed by each registration method) from preoperative to intraoperative space. Figure 42(b) depicts a transformed point set. The second point set is extracted from the intraoperative volume in Figure 42(c). Then, a Hausdorff Distance (HD) metric [47] is employed as the measurement of the degree of displacement between the two point sets.

Table 12 presents the results of this quantitative evaluation. A smaller HD value indicates better registration $(\text{HD} \geq 0)$, so perfect registration would have an HD of 0. The ratio $= \text{HD}_\text{X}/\text{HD}_\text{A-PBNRR}$ indicates the degree to which the error of the A-PBNRR is lower than the error of the X method, where $\text{X} \in \{\text{RR}, \text{B-Spline}, \text{PBNRR}\}$. Adaptive-PBNRR achieved the smallest error in each individual case and the smallest average error (3.63 mm) among all four methods. In thirty test cases, Adaptive-PBNRR is 5.47, 4.34, and 5.06 times more accurate in the presence of resection than RR, B-Spline, and PBNRR,

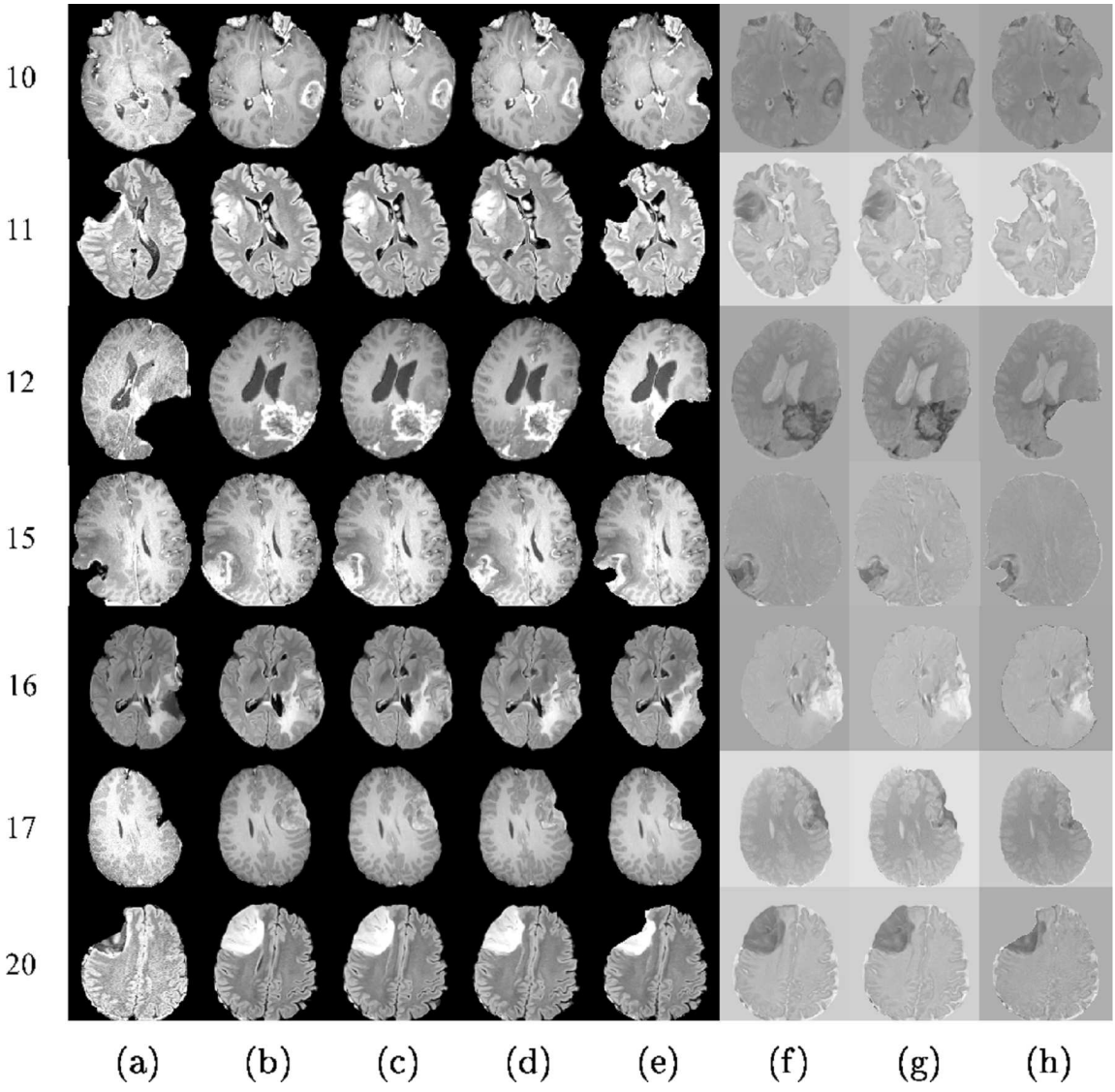


Figure 40: Qualitative results. Each row represents the same slice of a 3D volume for the case numbered on the left. Cases 10, 11, and 12 are partial tumor resections. Cases 15, 16, 17, and 20 are total tumor resections. From left to right: intraop MRI (a); deformed preop MRI after (b) rigid registration, (c) B-Spline, (d) PBNRR, and (e) A-PBNRR; difference between intraop MRI and (f) B-Spline, (g) PBNRR, and (h) A-PBNRR.

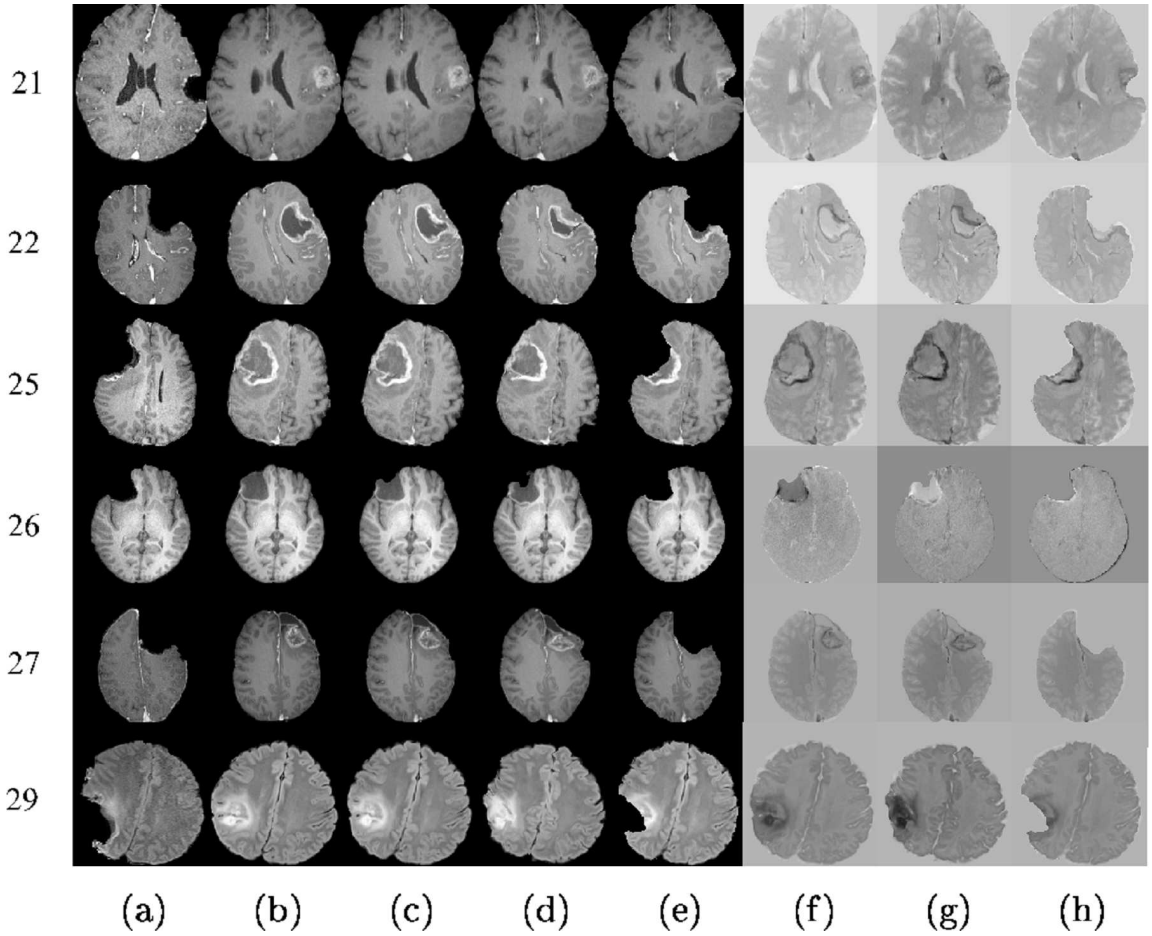
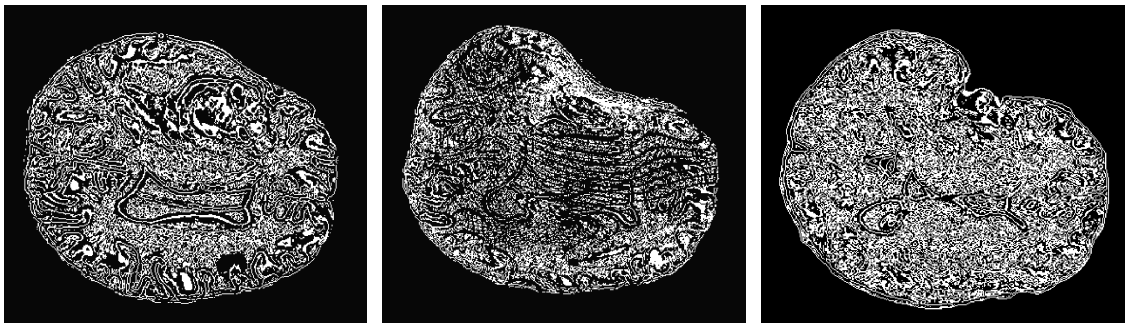


Figure 41: Qualitative results. Each row represents the same slice of a 3D volume for the case numbered on the left. Cases 21, 22, and 25 are total tumor resections. Cases 26, 27, and 29 are supra total tumor resections. From left to right: intraop MRI (a); deformed preop MRI after (b) rigid registration, (c) B-Spline, (d) PBNRR, and (e) A-PBNRR; difference between intraop MRI and (f) B-Spline, (g) PBNRR, and (h) A-PBNRR.

respectively. Note that this study utilized a 100% HD metric unlike our previous work [4], which featured a 95% HD metric. Figure 43 plots the HD error of data in Table 12.



(a) (b) (c)

Figure 42: Extracted Canny points in a single slice for quantitative evaluation of registration accuracy using the HD metric. (a): Points extracted from the preoperative MRI; (b): Points of (a) after transformation to the intraoperative space; (c): Points extracted from the intraoperative MRI. The HD metric is computed between point sets (b) and (c). Note that the Canny points are generally different from feature points used for registration.

3.8.3 QUANTITATIVE ASSESSMENT VIA ANATOMICAL LANDMARKS

Registration accuracy was quantitatively evaluated using anatomical landmarks selected by a neurosurgeon, as suggested in [87] (Figure 44). The neurosurgeon located six landmarks in each registered preoperative image volume and corresponding intraoperative image volume. Landmarks A and B were selected in the cortex near the tumor; C and D were selected at the anterior horn and the triangular part of the lateral ventricle, respectively; E and F were selected at the junction between the pons and mid-brain and at the roof of the fourth ventricle, respectively. For each landmark, the error was calculated as the distance between the landmark location in the registered preoperative image and the corresponding intraoperative image. Table 13 presents the minimum, maximum, and mean errors for each case. The assessment confirms that the Adaptive-PBNRR provides the most accurate registration, with an average minimum error of 1.03 mm and average mean error of 3.22 mm. In summary, Tables 12 and 13 suggest that independently of the evaluation method the Adaptive-PBNRR outperforms all three registration methods used in this evaluation. These quantitative results are consistent with the quality data presented in Figures 40-41 above.

Table 12: Quantitative registration results using the HD metric. HD_{RR}, $HD_{BSPLINE}$, HD_{PBNRR} and $HD_{A\text{-}PBNRR}$ are alignment errors after Rigid Registration (RR), B-Spline, PBNRR, and A-PBNRR registration, respectively. HD are in mm. The number in parenthesis denotes the number of adaptive iterations for A-PBNRR. BS: Brain Shift; PR: Partial Resection; TR: Total Resection; STR: Supra Total Resection.

Case	Type	HD_{RR}	$HD_{BSPLINE}$	HD_{PBNRR}	$HD_{A\text{-}PBNRR}$	$\frac{HD_{RR}}{HD_{A\text{-}PBNRR}}$	$\frac{HD_{BSPLINE}}{HD_{A\text{-}PBNRR}}$	$\frac{HD_{PBNRR}}{HD_{A\text{-}PBNRR}}$
1	BS	11.07	9.30	7.63	3.48(2)	3.18	2.67	2.19
2	BS	24.64	24.51	21.39	2.77(5)	8.90	8.85	7.72
3	BS	10.49	7.75	10.53	5.88(3)	1.78	1.32	1.79
4	BS	6.59	6.51	4.97	2.64(2)	2.50	2.47	1.88
5	BS	7.68	5.28	5.73	2.65(2)	2.90	1.99	2.16
6	BS	8.54	8.54	5.55	3.48(2)	2.45	2.45	1.59
7	BS	8.99	8.99	7.36	4.33(3)	2.08	2.08	1.70
8	PR	17.00	17.00	16.49	5.69(4)	2.99	2.99	2.90
9	PR	10.59	5.28	10.76	2.30(3)	4.60	2.30	4.68
10	PR	16.15	13.78	15.12	4.60(7)	3.51	3.00	3.29
11	PR	26.89	15.86	26.89	4.00(6)	6.72	3.97	6.72
12	PR	29.93	21.34	27.76	2.83(7)	10.58	7.54	9.81
13	TR	25.51	25.18	22.50	4.97(4)	5.13	5.07	4.53
14	TR	5.59	5.59	3.43	3.09(1)	1.81	1.81	1.11
15	TR	17.90	16.94	15.56	4.11(9)	4.36	4.12	3.79
16	TR	18.85	17.49	17.38	3.57(3)	5.28	4.90	4.87
17	TR	17.14	7.48	15.41	4.25(2)	4.03	1.76	3.63
18	TR	25.72	25.72	23.90	3.42(6)	7.52	7.52	6.99
19	TR	25.43	17.63	25.22	3.30(9)	7.71	5.34	7.64
20	TR	23.61	21.42	22.89	3.66(4)	6.45	5.85	6.25
21	TR	19.24	14.61	19.89	2.40(6)	8.02	6.09	8.29
22	TR	30.37	21.39	28.96	3.13(7)	9.70	6.83	9.25
23	TR	15.16	11.89	13.96	3.15(4)	4.81	3.77	4.43
24	TR	13.47	8.90	13.66	3.28(4)	4.11	2.71	4.16
25	TR	23.22	14.99	21.44	3.08(9)	7.54	4.87	6.96
26	STR	17.59	17.12	16.63	4.19(5)	4.20	4.09	3.97
27	STR	35.72	27.77	33.57	3.71(8)	9.63	7.49	9.05
28	STR	32.32	29.43	30.13	3.45(6)	9.37	8.53	8.73
29	STR	18.48	13.30	18.15	3.97(4)	4.65	3.35	4.57
30	STR	27.07	15.55	24.91	3.54(7)	7.65	4.39	7.04
Average		19.03	15.22	17.59	3.63	5.47	4.34	5.06

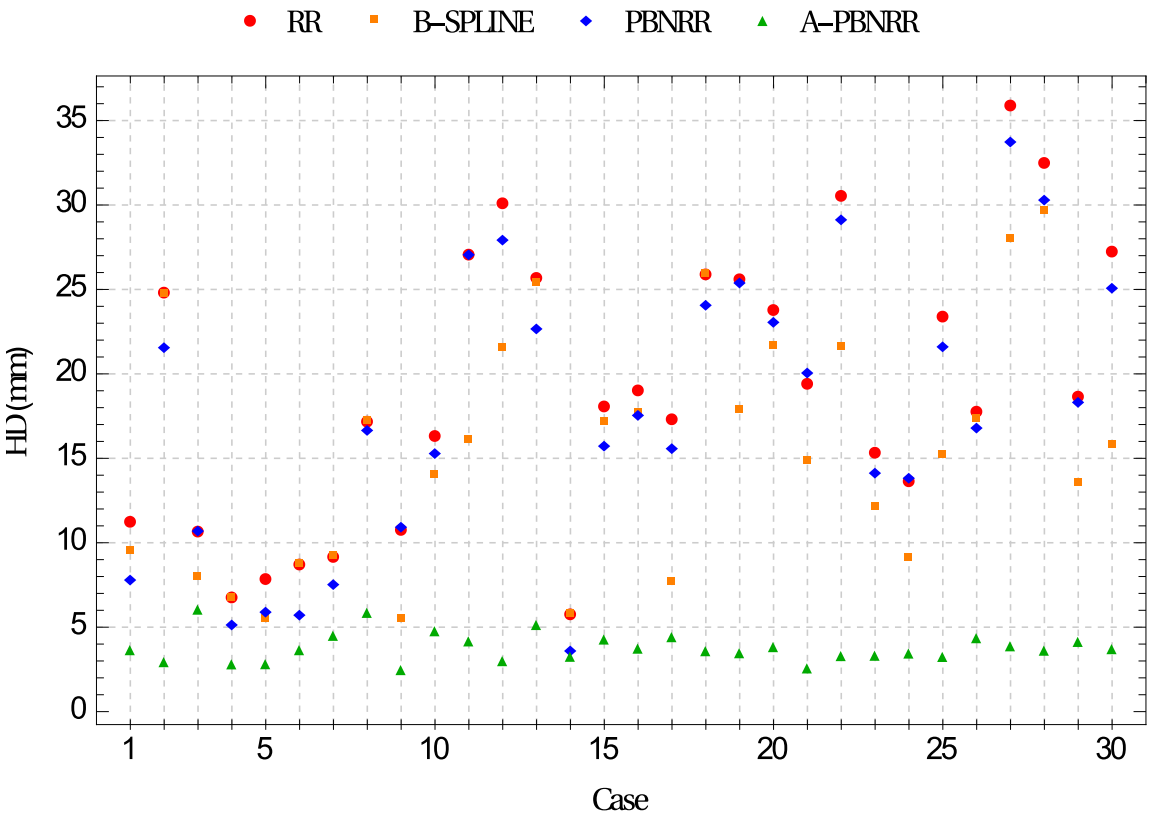


Figure 43: Plot of Hausdorff Distance (HD) errors of Table 12. Brain shift: cases 1-7; partial resection: cases 8-12; total resection: cases 13-25; supra total resection: cases 26-30.

3.8.4 PERFORMANCE

All methods were parallelized for shared memory multiprocessor architectures. Figure 45 presents the end-to-end execution time for the registration of preoperative to intraoperative images for all thirty cases. Rigid registration, B-Spline, PBNRR, and A-PBNRR required on average 0.84, 8.98, 0.83, and 1.42 minutes, respectively (including I/O). Note that the B-Spline method is the most computationally intensive, requiring more than 8 minutes in 17 out of 30 cases. A different set of B-Spline parameters, such as a smaller sampling percentage, a smaller number of histogram bins, or a coarser grid, could potentially improve performance at the cost of accuracy. Although A-PBNRR is slower than RR and PBNRR, it has significantly better accuracy in the presence of resection than the other

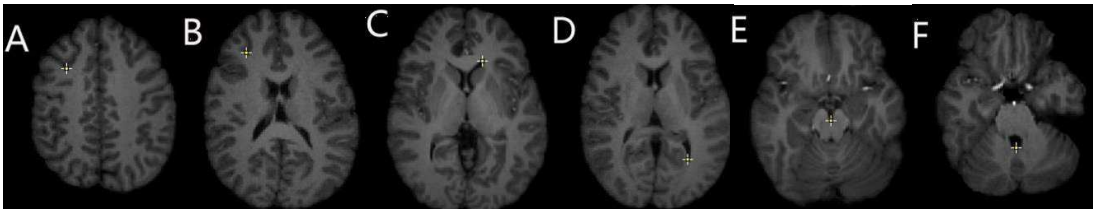


Figure 44: Anatomical landmarks (A-F) used for quantitative evaluation of registration accuracy. A neurosurgeon located the landmarks. A, B: cortex near tumor; C: anterior horn of lateral ventricle; D: triangular part of lateral ventricle; E: junction between pons and mid-brain; F: roof of fourth ventricle.

Table 13: Quantitative registration results using six anatomical landmarks (A-F). The values are the average minimum, maximum, and mean errors computed over thirty cases for Rigid Registration (RR), B-Spline, PBNRR, and A-PBNRR registration.

Method	Average min error (mm)	Average max error (mm)	Average mean error (mm)
RR	3.19	8.90	5.60
BSPLINE	2.15	8.29	4.40
PBNRR	1.11	6.81	3.47
A-PBNRR	1.03	6.59	3.22

methods, and it is fast enough to satisfy the constraints of image-guided neurosurgery, where registration times of less than two to three minutes are desired.

Figure 46 presents scalability results for the registration of preoperative to intraoperative images for all thirty cases. The speedup was computed as the ratio of the time taken by the sequential implementation to the time taken by the parallel implementation using 12 hardware cores. PBNRR exhibits poor scalability because it includes only one parallel module (block matching). A-PBNRR demonstrates good scalability (up to 6.12) due to a much higher parallelization fraction.

3.9 DISCUSSION AND FUTURE WORK

Recent advances in neuroimaging such as fMRI and DTI allow neurosurgeons to plan tumor resections that minimize damage to eloquent cortical regions and white matter tracts [200, 17, 102, 153, 189]. A number of commercial systems (e.g., Brainlab Curve Image-Guided Therapy system) can register fMRI and DTI to preoperative anatomical

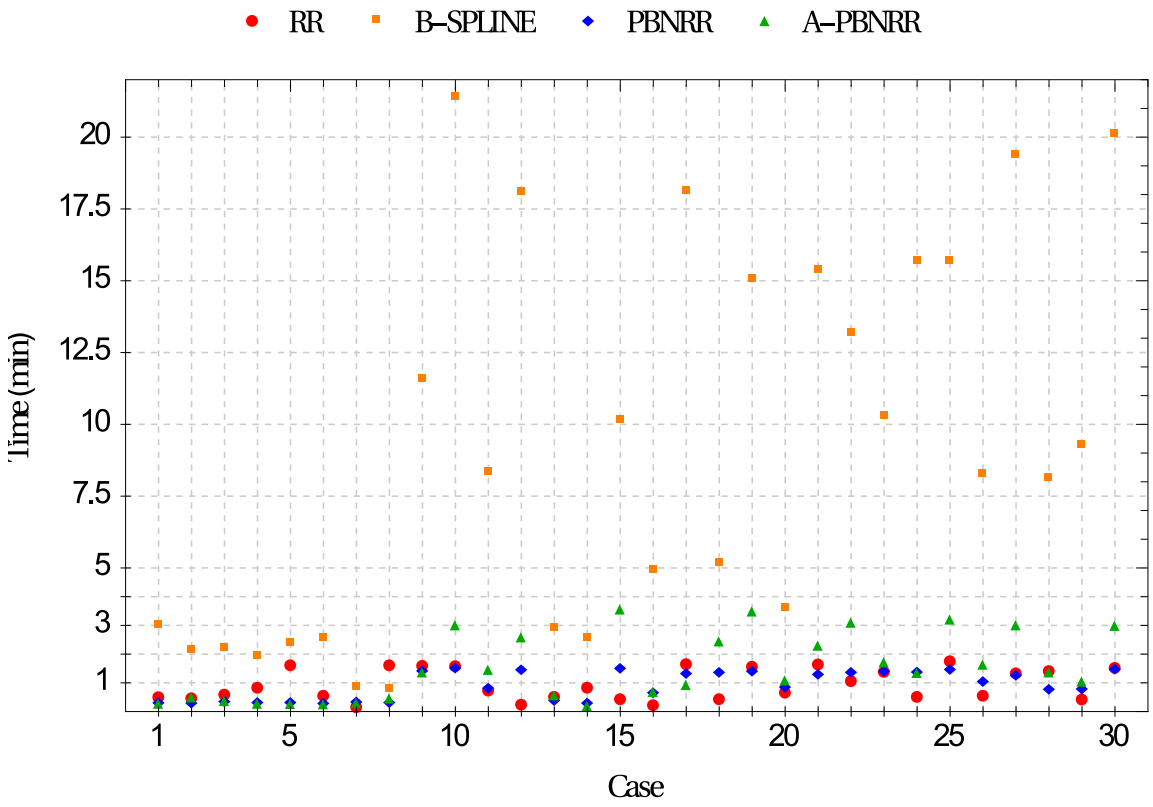


Figure 45: End-to-end execution times for registration from preoperative to intraoperative images. All registration methods were run in parallel on 12 hardware cores on a DELL workstation with 12 Intel Xeon X5690@3.47 GHz CPU cores and 96 GB of RAM. Execution times include I/O. The mesh generation time is excluded from the PBNRR (preoperative step) but is included in the A-PBNRR (intraoperative step).

MRI images and then map this data to the patient intraoperatively using rigid registration. However, significant deformation of the brain is possible during surgery, especially in the presence of tumor resection, making rigid registration insufficient and surgical plans made with preoperative data invalid [149].

Intraoperative MRI can be used to observe the deformed brain during surgery. While it is impractical to acquire fMRI and DTI intraoperatively, the preoperative MRI image can be registered to an intraoperative MRI image using non-rigid registration. The resultant registration can then be applied to the preoperative fMRI, DTI, and the surgical plan, providing more accurate, updated guidance to the neurosurgeon [4]. Figure 47 depicts such a

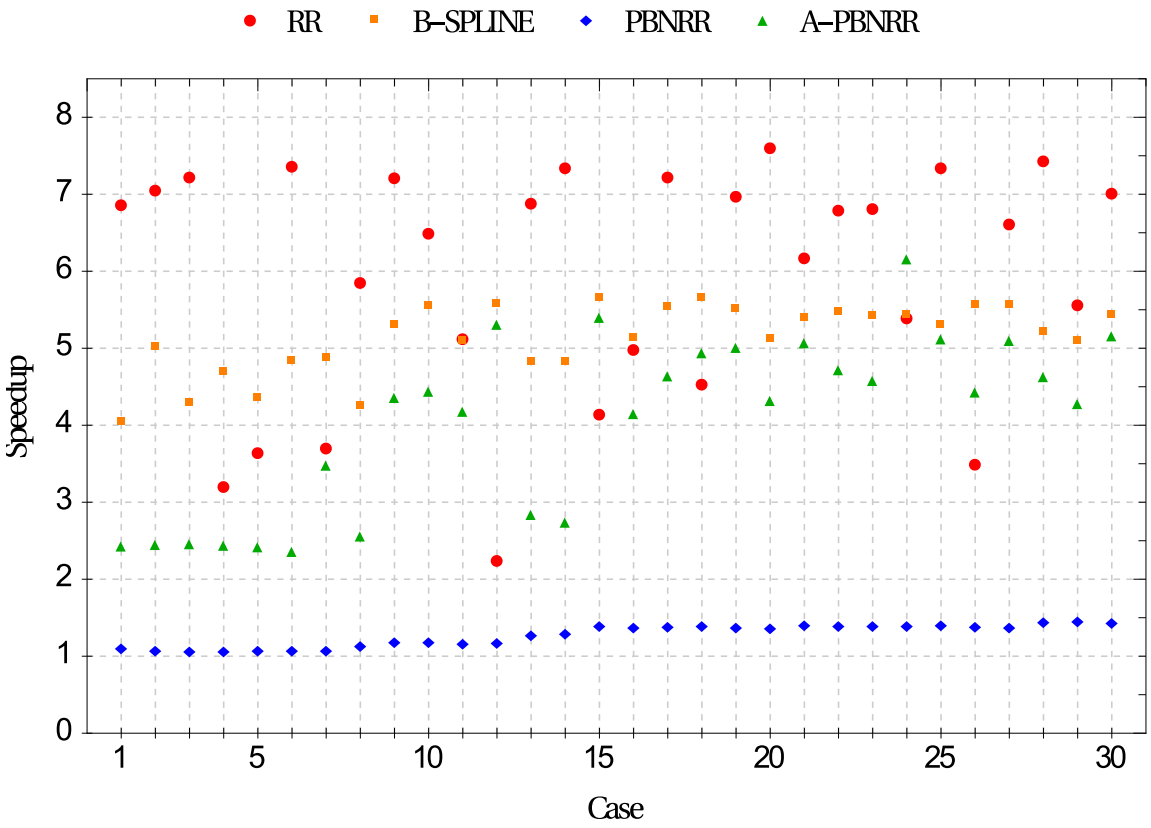
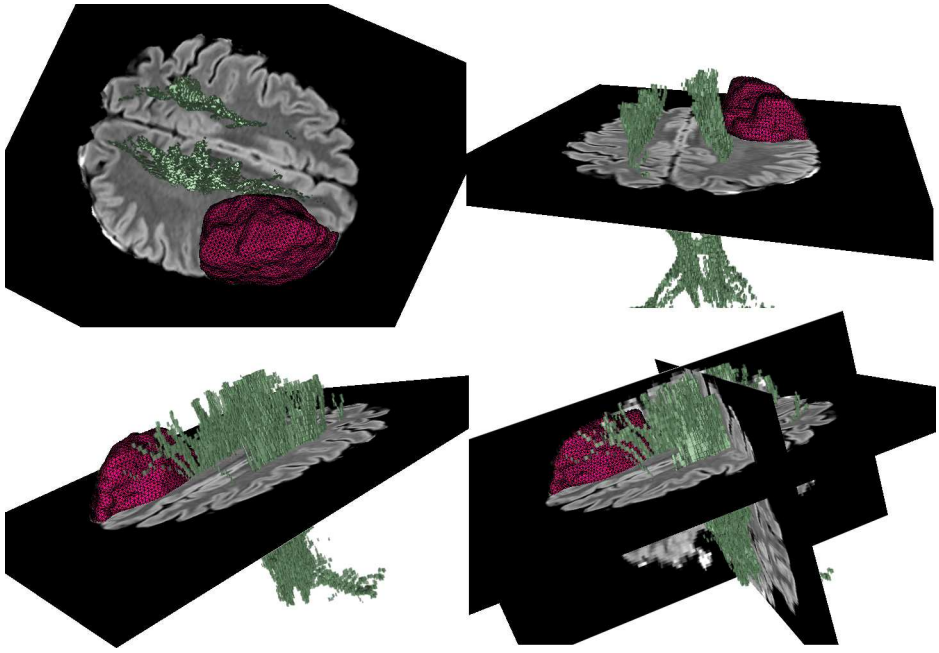


Figure 46: Scalability for registration from preoperative to intraoperative images using a DELL workstation with 12 Intel Xeon X5690@3.47 GHz CPU cores and 96 GB of RAM. Execution times include I/O.

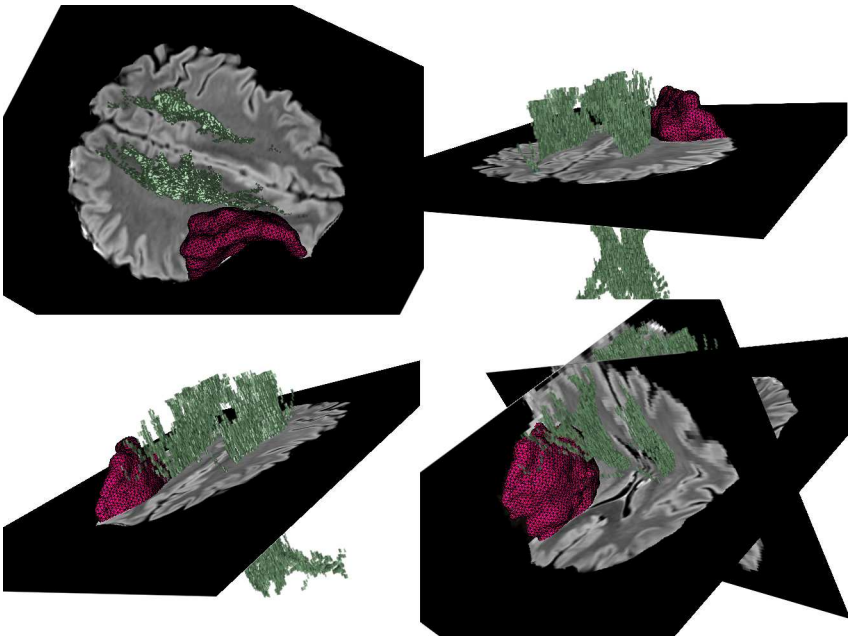
visualization.

Non-rigid registration algorithms remain computationally expensive and have proven to be impractical for use in clinical settings in the past. Thus, one of the goals of this research has been to employ parallel and distributed computing architecture to provide faster and more effective registration for image-guided neurosurgery [41, 4].

In this study, four methods for registering preoperative to intraoperative MRI images were compared among thirty glioma cases from two different hospitals; a rigid registration implemented in 3D Slicer v4.4.0 [98], a B-Spline non-rigid registration implemented in 3D Slicer v4.4.0 [98], a physics-based non-rigid registration (PBNRR) implemented in ITKv4.7.0 [120], and a modification of PBNRR that handles tumor resections (Adaptive-PBNRR) [57]. Many of these cases involved the resection of significant tumor volumes.



(a)



(b)

Figure 47: Rigid (a) and non-rigid (b) alignment of preoperative DTI tractography (the motor pathway volume is exported to DICOM format) with intraoperative T1 FLAIR volume from patient 20 (male, 36 years old, grade II glioma). The DTI tractography is shown together with a tumor model (red) during the neurosurgical resection. Part of the tumor has been resected. Visualizing the tumor model and DTI tractography in the surgical context helps the neurosurgeon to achieve an appropriate volumetric resection.

Three methods were employed in assessing the registration accuracy: (i) visual assessment, (ii) a Hausdorff Distance-based metric, and (iii) a landmark-based approach using anatomical points identified by a neurosurgeon. The processing speeds of these methods were also compared.

The experimental results confirm that, of the four methods, Adaptive-PBNRR provides the most accurate registration, with an average error of 3.2 to 3.6 mm for landmark-based and Hausdorff Distance-based metrics, respectively, for anisotropic image spacing (e.g., $0.488 \times 0.488 \times 1.0\text{mm}^3$, Table 6). The extent of the resection (partial, total, or supra total) does not significantly affect this accuracy (Figures 40-41 and 43, Table 12). Performance analysis shows that Adaptive-PBNRR is sufficiently fast to be useful in a clinical setting.

The ultimate goal is to provide accurate, high-quality data to neurosurgeons intraoperatively to allow them to make the best possible decisions during tumor resection. This work has shown through the study of thirty patients that we can map preoperative image data to the patient with an average error of a few millimeters and with computation times that are acceptable in a clinical environment. Future efforts will continue this focus on improving registration accuracy and decreasing computation times, using Cloud computing and Machine Learning. Preliminary results for the PBNRR method suggest a reduction in the registration error in the order of 30% to 50% [141]. Decreasing computation times will require exploring ways to improve parallelization of registration methods. Improving accuracy will require an investigation into higher order finite element modeling to study the impact on accuracy and performance. Additionally, the effect of segmentation and model construction on the registration accuracy remains a potential area of future study. Specifically, an investigation into the effect of higher quality segmentation of the brain surface and structures that constrain brain deformation, such as the skull cavity, the falx cerebri, and the tentorium cerebelli, would reveal the impact of incorporating more tissues, including blood vessels and the ventricles, into the brain model. Finally, because intraoperative MRI is not available in many neurosurgical suites, further investigation incorporating the use of intraoperative ultrasound to track brain deformation during surgery would extend these registration methods to map preoperative image data to intraoperative ultrasound.

3.10 CONCLUSION

This study compared four methods for registering preoperative image data to intraoperative MRI images in the presence of significant brain deformation during glioma resection in thirty patients. The Adaptive Physics-Based Non-Rigid Registration method developed

in this thesis extends earlier work on physics-based finite element methods by using real-time multi-tissue mesh adaptation to automatically remove elements in the area of the resected tumor, thereby automatically handling deformation in the presence of resection. The proposed algorithm exploits additional parallelism over existing physics-based methods with corresponding performance improvements so that, even with multiple iterations, it requires on average less than two minutes to perform non-rigid registration. Both the registration accuracy and performance were found to be of clinical value in the operating room.

CHAPTER 4

HIGH QUALITY PARALLEL UNSTRUCTURED GRID GENERATION FOR FINITE ELEMENT SIMULATIONS

4.1 INTRODUCTION

This chapter presents a new parallel unstructured grid generator for efficient discretization of piecewise linear complex domains in Computational Fluid Dynamics (CFD) modeling using Finite Elements (FE). FE procedures [15] are useful to analyze the robustness and performance of components and assemblies in engineering and to validate and optimize products and manufacturing in industry.

To the best of our knowledge, this is the first method that optimizes the grid connectivity in parallel throughout the generation procedure, including a post-improvement step. The connectivity optimization is based on a speculative (optimistic) approach that has been proven to perform well on hardware-shared memory (i.e., single chip) [40, 70]. Our new speculative paradigm repetitively reconnects the grid using tightly-coupled topological transformations. Simple hill-climbing is employed to maximize the quality of the worst local element. The local reconnection scheme is based on: (i) data over-decomposition, (ii) atomic operations to avoid data races and maintain a valid grid throughout the procedure, and (iii) load-balancing to redistribute work-units among the threads. The grid generation engine is essentially a combination of Advancing Front type point placement [123, 127], direct point insertion [132], and parallel multi-threaded connectivity optimization schemes [171, 201].

The framework developed in this thesis is a complete end-to-end solution that focuses on improving both the quality and the performance. Therefore, it can be used for academic or industrial purposes. The design and implementation of this framework fulfills to a high degree the requirements of an ideal grid generator:

1. **Stability**. The quality of the grid generated in parallel must be comparable to that of a grid generated sequentially. The quality is defined in terms of the shape of the elements (using a chosen space-dependent metric) and the number of the elements (fewer is better for the same shape constraint).

2. **Robustness**. The ability of the software to correctly and efficiently process any input data and to consistently generate high quality elements. Operator intervention into a parallel computation is not only highly expensive, but most likely infeasible due to the large number of concurrently processed sub-problems.

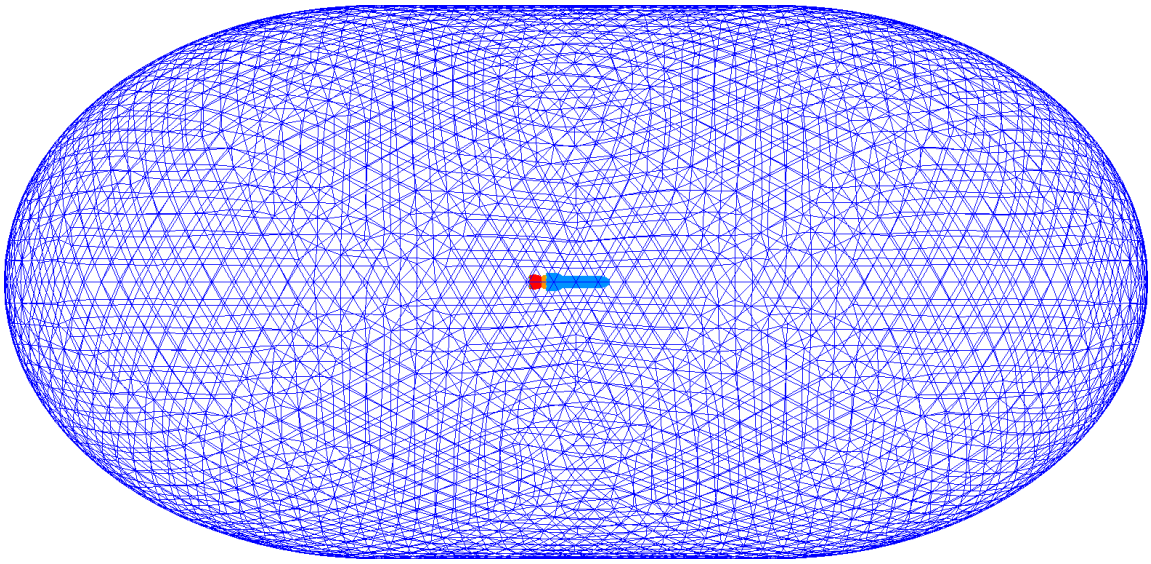
3. **End-User Productivity**. The ability of the software to process the input data faster compared to state-of-the-art codes. Scalability is defined as the ratio of the time taken by the best sequential implementation to the time taken by the parallel implementation. The speedup is always limited by the inverse of the sequential fraction of the software, and therefore all non-trivial stages of the computation must be parallelized to leverage the current multi-core architectures.

4. **Code Re-Use**. A modular design of the parallel software, such that it can be replaced and/or updated with minimal effort. Due to the complexity of grid generation codes, this is the only practical approach for keeping up with the ever-evolving algorithms and computer architectures.

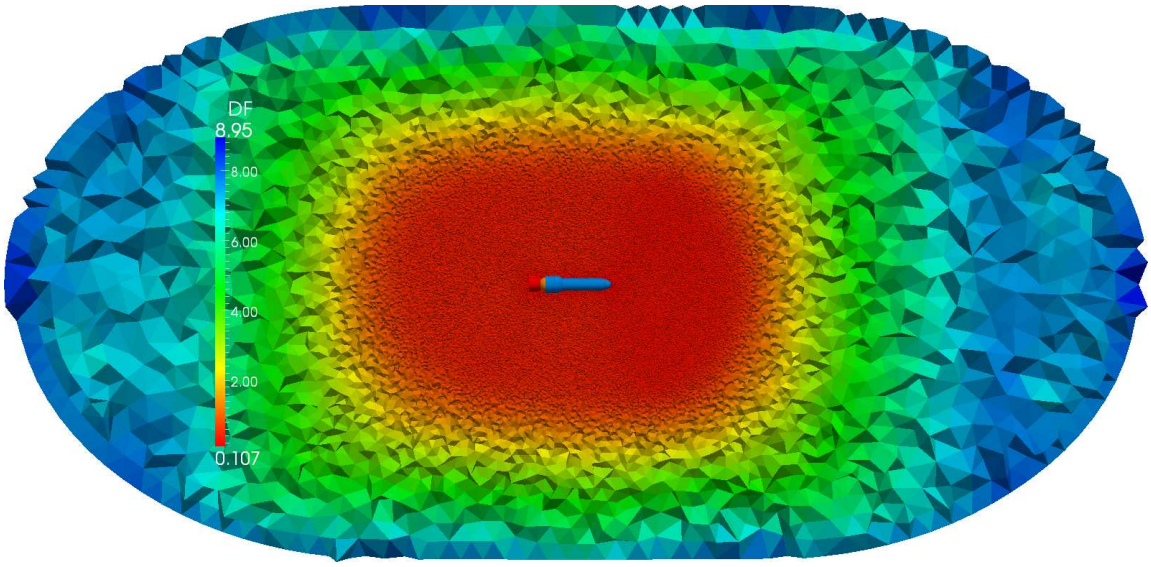
Given a boundary surface grid of the domain to be discretized, the proposed method first constructs an initial volume grid that conforms to the input boundary grid. The initial grid might contain few points in the volume due to the fact that a tetrahedralization of an arbitrary polyhedron without interior points is not always possible [165, 31]. New points are created with an Advancing Front type point placement. Points are inserted into the grid by directly subdividing the tetrahedra that contain them. The connectivity is then optimized using parallel speculative local-reconnection. The generation procedure stops when a desired point distribution function is satisfied. The remaining bad shape elements are eliminated in a post-processing step that utilizes various mechanisms, including a parallel speculative local-reconnection. The shape of the elements is critical for the accuracy and the convergence of a finite element solution. For example, elements with large dihedral angles tend to increase the discretization error in the finite element solution [8]. On the other hand, elements with small dihedral angles are bad for matrix conditioning but not for interpolation or discretization [74, 173]. Figure 48 depicts a volume grid generated by the proposed method.

4.2 RELATED WORK

Two approaches are typically used to generate a tetrahedral grid from an input surface grid: Delaunay or Advancing Front [190]. Delaunay methods can handle constrained



(a) Input surface grid



(b) Cut of the output volume grid ($\approx$ 200 M tetrahedra)

Figure 48: Cut of unstructured tetrahedral grid of flow field of a rocket. The grid is generated with the parallel optimistic algorithm developed in this thesis. The point Distribution Function (DF) varies between the domain boundaries. The input surface grid obtained from Center of Advanced Vehicular Systems in MSU.

polyhedral domains of arbitrary complexity with the use of heuristic mesh modification techniques [79, 10]. Other Delaunay approaches provide mathematically guarantees on the grid quality but do not preserve the boundary triangulation [177, 146].

The first efforts on the parallelization of existing sequential Delaunay grid generation algorithms are reported in [44, 152]. These methods are based on the parallelization of a Bowyer-Watson kernel [24, 195]. A tightly-coupled distributed parallel Delaunay refinement algorithm for simple polyhedral domains whose constituent bounding edges and surfaces are separated by angles between 90° to 270° is presented [147]. This algorithm, can create and place large meshes up to 6 times faster than the traditional approach (i.e., sequentially generate a sufficiently dense mesh, partition the mesh into submeshes, distribute the submeshes to the processors, and sequentially refine the mesh) to generating and refining a distributed mesh.

A Delaunay method that allows for safe insertion of points independently without synchronization is presented [37]. This method, based on a carefully constructed octree, splits the worklist of the candidate points up into smaller lists such that the available sequential codes can be used without modifications to process the sublists. The drawback is that the amount of work assigned to different processors can vary significantly. However, by using over-decomposition in a combination with runtime software system for dynamic load balancing [12], the work among the nodes can be redistributed.

In some cases (e.g., high-aspect ratio viscous grids or three-dimensional grids with sliver elements formed from four nearly coplanar points) the Delaunay criterion may not be the most suitable. For this reason, other approaches to improve the quality over that of a Delaunay grid by using topological transformations have been introduced [14, 114, 97, 178].

Advancing Front methods offer the advantages of a high quality grid due to optimal point placement and boundary integrity [127, 123]. Its main drawbacks are complexity and lower element quality when fronts collide in 3-dimensions. An Advancing Front grid generator for shared memory architectures is presented in [124]. The domain to be gridded is first subdivided spatially using a coarse octree, and then the boxes are gridded in parallel. In [94], a coarse tetrahedral mesh is generated first to provide the basis of block interfaces and then partitioned into a number of subdomains using METIS [100] partitioning algorithms. A volume grid is generated on each subdomain in parallel using an Advancing Front method, and all subdomains are later combined to create a single grid. To remove the artifacts in the interfaces between subdomains, an angle-based node smoothing was

used, but no topological changes (i.e., local reconnections) are introduced.

Grid generation codes typically improve the shape of the elements in a post-processing step, called *grid optimization* or *grid quality improvement* [50, 71]. On the contrary, the proposed method optimizes the connectivity throughout the generation procedure (including a post-improvement step) to obtain a maximum quality. A combined quality criterion is utilized to optimize the connectivity: Delaunay in-sphere [51] and Min-Max type [14]. The shape of the elements is not always improved if the combined criterion is used only on the final grid as a post-processing step.

Commonly, a grid optimization algorithm incorporates three mechanisms: (i) vertex relocation (smoothing) [28, 71, 132, 109], (ii) local reconnection with topological transformations [73, 178, 109, 50, 97, 132, 179], and (iii) point insertion/deletion to localy refine/coarsen the grid [50, 109, 132, 179]. This thesis implements efficiently all these mechanisms. Topological transformations are fundamental operations in local reconnection. A topological transformation removes elements and replaces them with a different set of elements that occupy the same space. Transformations can be computationally expensive and time-consuming, mainly for two reasons: (i) a transformation does not always produce a geometrically valid connectivity, hence the orientation of the new elements needs to be verified, and (ii) a quality metric needs to be computed to decide whether the new connectivity is locally optimal or not. For those reasons, this thesis implements a parallel speculative local reconnection approach, therefore significantly reducing the execution time. Experimental evaluation of our method indicates that a sequential optimization with local reconnection accounts for at least 70% of the grid generation time (without post-improvement).

Many parallel Delaunay or Advancing Front algorithms have been proposed [42, 70, 162, 37, 147, 166, 125, 124]. However, few parallel local reconnection algorithms have been presented in the literature. In [126] a parallel distributed Advancing Front grid generation method with quality improvement is presented. Quality improvement includes a combination of several algorithms, i.e., diagonal swapping, removal of bad elements, node smoothing, and selective grid movement. In the first pass of quality improvement, each subgrid is reconnected by swapping edges in its interior, so the inter-processor boundary remains unchanged. In the second pass, a new partition is created by adding 1-2 extra layers of elements to each subdomain from the neighboring domains, the sub-grids are redistributed among processors, and grid improvement operations are performed again. However, it is unclear how the work load is balanced after re-partitioning.

A multi-threaded local reconnection and smoothing algorithm for grid improvement is presented [171]. The parallelization of smoothing operations is based on an existing data-decomposition technique [73], which colors the dual graph of the grid to subdivide the points into a few independent sets. The parallelization of local reconnection operation is based on a new data-decomposition technique, which defines a feature point in the interior of each local reconnection operation and sorts the feature points along a Hilbert curve [22, 179]. The decomposition of this Hilbert curve results in an initial load-balanced distribution of local operations, i.e., edge removals from those low quality tetrahedra.

A two-step thread-parallel edge and face swapping algorithm is presented [201]. In the first step, a vertex locking strategy is introduced to select a maximal conflict-free set of edges and faces for swapping. In the second step, edge and face migration between threads is performed to balance the work-load in each thread, and then parallel edge and face swaps are applied without any interference. However, the evaluation of the 3-dimensional algorithm is limited.

Other approaches suggest a combination of smoothing and untangling operations [18, 106]. A log-barrier interior point method is developed to solve a smooth constrained optimization problem and untangle a mesh with inverted elements, improving its quality [168]. This method is parallelized for distributed memory machines using an edge-based coloring communication synchronization technique in which edges corresponding to a graph of communicating processes are colored to synchronize the communication [167].

A parallel optimization technique that smoothens independent sets of vertices simultaneously is developed in [71]. This technique performs local vertex movement using a vertex-coloring scheme to avoid conflicting updates to vertex positions. The method is implemented for a parallel random access machine (PRAM) model and distributed memory architectures.

An optimization-based smoothing algorithm for anisotropic mesh adaptivity is presented in [83]. The smoothing kernel solves a non-linear optimization problem by differentiating the local mesh quality with respect to mesh vertex position and employing hill-climbing to maximize the quality of the worst local element. The method is parallelized for hybrid OpenMP/MPI using standard coloring techniques.

Previous work on 3-dimensional Delaunay refinement [70] indicates that a speculative approach performs well on hardware-shared memory. Similarly to a Delaunay reconnection, roll-backs are possible during speculative reconnection with topological transformations, due to intersections of polyhedra that are attempted to transform concurrently. The

proposed scheme employs atomic operations to avoid such intersections, thus maintaining a valid connectivity throughout the procedure. To exploit additional parallelism, our method implements a load-balancer to migrate work-units (buckets) from busy threads to threads without work. The granularity of the decomposition (i.e., size of work-units) and the amounts of data for migration can be adjusted to achieve an optimum performance.

The number of cores to emerging chips and for High Performance Computing (HPC) nodes is expected to be in the order of hundreds. Therefore, in future efforts, a Parallel Data Refinement (PDR) layer [34, 35, 37] will be implemented on top of the proposed speculative approach to improve the scalability by at least a factor of $O(10^2)$. Our ultimate goal is to implement a telescopic approach for current and emerging parallel architectures with several layers in network and memory hierarchy, therefore achieving extreme-scale adaptive simulations on the complex, heterogeneous HPC architectures that will be increasingly prevalent through 2030 [180, 43] (Figure 49).

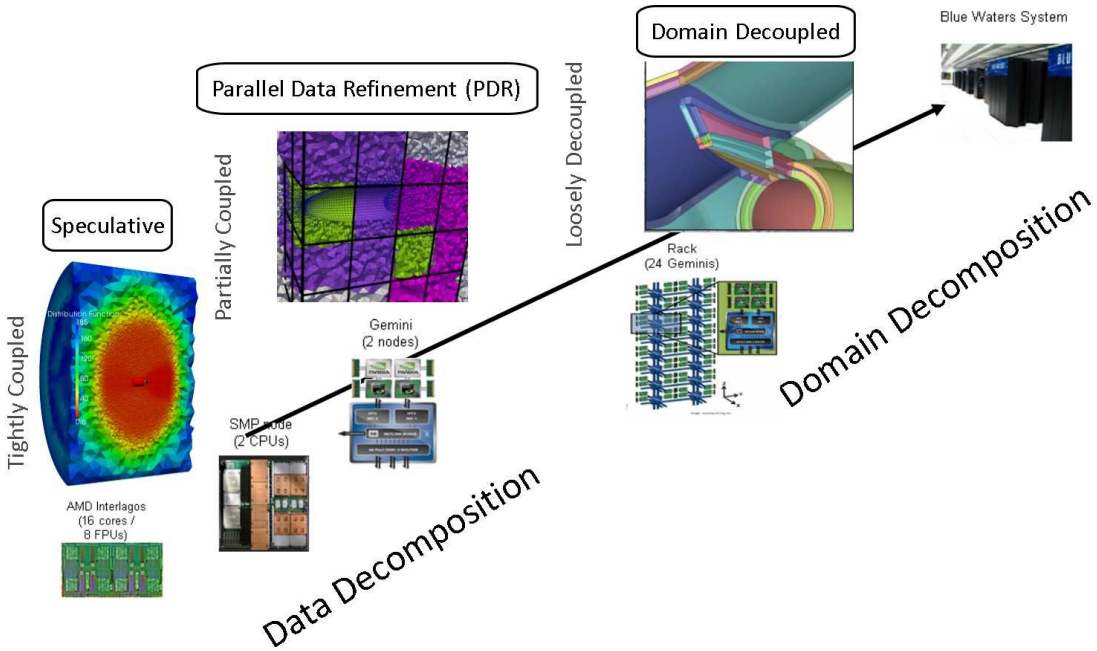


Figure 49: Telescopic Approach for Extreme-Scale Parallel Mesh Generation [43]

The proposed algorithm is capable of efficiently generating billions of elements for realistic aerospace configurations. It exhibits "ultimate" robustness[1] on recovering boundary surfaces with bad shape or high aspect-ratio anisotropic triangles. Both the sequential and the parallel implementation of this code generates grids of a comparable element quality

[1] this term was first used in [78]

with state-of-the-art technology [132], currently incorporated in several commercial systems (i.e., DoD CREATE-MG Capstone, Lockheed Martin/DoD ACAD, Boeing MADCAP, MSU SolidMesh, and Altair HyperMesh). Experimental evaluation results in this thesis indicate that our method completes the grid refinement up to 2.5 and 10 times faster compared to state-of-the-art code [132], when 1 and 12 hardware cores are utilized, respectively. The parallel speculative connectivity optimization scheme exhibits near-linear scalability when 24 cores are utilized to discretize the flow domain of a DLR-F6 Airbus aircraft with approximately 1.5 billion elements.

Details of the proposed algorithm are presented in the following sections along with an extensive evaluation on quality and scalability aspects of the code for 3-dimensional aerospace configurations.

4.3 GRID GENERATION STEPS

The developed grid generation software is called *CDT3D*. It takes as an input a Piecewise Linear Complex (PLC) [140, 175] of the domain to be discretized. Typically, the PLC is a boundary surface triangulation of the domain. The output is a boundary-conforming isotropic tetrahedral grid with an element size that is defined by a point distribution function. The end-to-end procedure includes three basic steps and a final post-improvement step:

1. **Delaunay Tetrahedralization**. A initial grid of 6 tetrahedra is created that encloses the input boundary points. The input points are consecutively inserted into the grid. After each insertion the grid is reconnected using a Delaunay empty sphere criterion.

2. **Boundary Recovery**. The Delaunay grid is modified to contain all the entities (edges and faces) of the input surface triangulation. The missing entities are recovered with topological transformations and edge/face partitioning. Additional points (called *Steiner* points) might be necessary to recover the input surface.

3. **Refinement**. New points are inserted into the grid until a desired point distribution function is satisfied. New points are created with an Advancing Front type point placement. Points are inserted into the grid by directly subdividing the tetrahedra that contain them. The connectivity is optimized in parallel with a combination of topological transformations. The refinement stops when no new points are created or accepted.

4. **Quality Improvement**. The grid is improved by vertex smoothing, parallel connectivity optimization, and point insertion with heuristics to remove isolated bad shape elements.

Details of these steps in the procedure are presented in the following sections.

4.4 DELAUNAY TETRAHEDRALIZATION

A Delaunay triangulation of a d-dimensional point set P is a d-dimensional simplicial complex D such that every simplex in D has the empty sphere property [51]. In 3-dimensions, each tetrahedron has a circumscribed sphere that contains no other point of P, and the underlying space $|D|$ is the convex hull of P. If no $d+2$ vertices of P share a common sphere, then the set of all Delaunay simplices uniquely forms the Delaunay triangulation of P.

Two well-known algorithms for constructing Delaunay tetrahedralizations are the Bowyer-Watson algorithm [24, 195] and the incremental flip algorithm [62]. Parallel tightly-coupled methods for Delaunay insertion and refinement [147, 70], as well as a partially-coupled and a decoupled method for distributed-memory systems based on Medial Axis decomposition [117, 36], were also developed. We implement a sequential Bowyer-Watson algorithm since this step is used only for the insertion of the input boundary points. The evaluation results indicate that this step accounts for less than 1% of the total grid generation time (Table 18) if the points are pre-sorted before insertion [179]. Note that the grids generated in this study contain a number of surface points which is approximately equal to the number of volume points to the 2/3 power.

Prior to the insertion of the input points P, an initial Delaunay grid enclosing P is constructed. The initial grid contains five tetrahedra and eight vertices. Before a point p is inserted into the grid a search is necessary to find the containing element. This operation is called point location. CDT3D implements a stochastic walk algorithm [52, 179] to locate p, and robust geometric predicates with dynamic filters for orientation and in-sphere tests [174]. After p is located, a cavity $C(p)$ is computed, consisting of the elements whose circumsphere contains p. These elements are deleted (because they violate the Delaunay property), and p is connected to the vertices on the boundary of $C(p)$.

Point location can be the bottleneck of the performance of an incremental Bowyer-Watson algorithm [121]. CDT3D improves the speed by: (i) pre-sorting P such that points geometrically close in space are likely close in their insertion order, and (ii) starting the search for the containing element from the latest created element. The same approach with

TetGen [179] is followed to pre-sort point set P. At first, the points are grouped using a Biased Randomized Insertion Order (BRIO) [2] and then ordered within each group along a Hilbert curve [22].

4.4.1 EXAMPLES ON DELAUNAY TETRAHEDRALIZATION

The performance of the Delaunay Tetrahedralization is evaluated on three geometries (Figure 50). CDT3D is compared to TetGen [179], a publicly available software for generating high quality tetrahedral meshes aimed to support numerical methods and scientific computing. Table 14 reports the results. CDT3D is approximately 15% slower than TetGen. This is because TetGen's data structures are highly optimized for sequential operations. On the other hand, CDT3D's data structures are designed to address both sequential and parallel operations. Small differences at the sorting timings occur for the same reason. In future efforts, this step can be parallelized using a tightly-coupled approach [70].

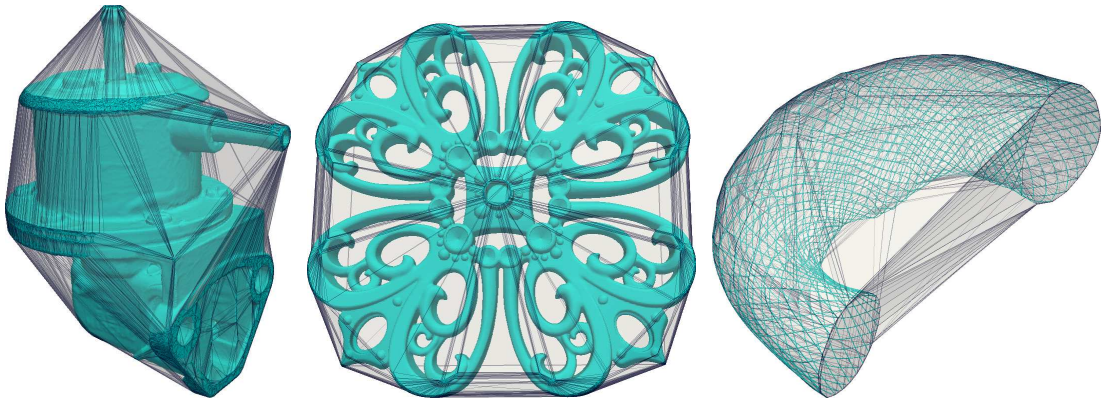
Table 14: Performance of the Delaunay tetrahedralization. CDT3D is compared with state-of-the-art open-source software TetGen v1.5.0 [179]. I/O time is not included in the results. The generation time without element sorting is reported in parenthesis. The experiments performed on a Linux machine with Intel Core i7-2600 CPU@3.40GHz, and 16 GB RAM.

Geometry	#Points	#Tets	Time (sec)			
			Sorting		Tetrahedralization	
			TetGen	CDT3D	TetGen	CDT3D
Oil-pump	3420144	3961092	0.94	0.99	7.32	8.29 (31.41)
Filigree	3086568	3515676	0.80	0.77	7.58	8.63 (20.80)
Flow Diverter	6103404	11544940	2.05	1.98	16.81	19.69 (55.43)

4.5 TOPOLOGICAL TRANSFORMATIONS

Topological transformations are necessary in the majority of grid generation algorithms [73, 132, 97, 50, 77, 178, 109]. Flips or swaps are alternative terms often used in computational geometry literature [62].

A transformation modifies the grid connectivity by removing elements and replacing them with a different set of elements that occupy the same space. Some types of transformations facilitate a point insertion or a point removal operation. A transformation is local if it involves a small number of elements. Topological transformations are typically used in conjunction with an objective function to optimize the grid. Typical objective functions



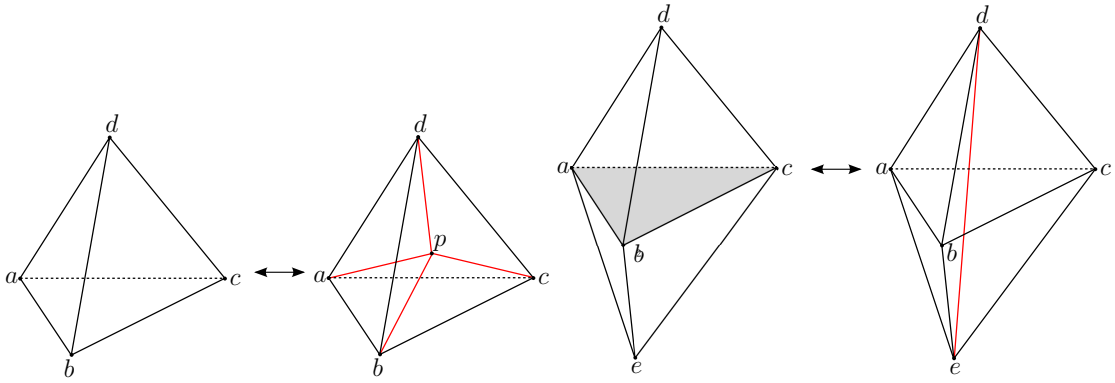
(a) oil-pump (b) filigree (c) flow diverter

Figure 50: Geometries to evaluate the performance of the Delaunay tetrahedralization. The input surface grids (cyan) are shown together with their convex hull (gray). (a) and (b) obtained from AIM@SHAPE Repository (http://shapes.aim-at-shape.net). (c) obtained from the Neurosurgery Department at Stony Brook University.

are the average quality of the elements or the quality of the worst element. CDT3D implements various objective functions such as: (i) Delaunay empty sphere property [51], (ii) maximization of the minimum Laplacian edge weight [14], (iii) minimization of the maximum dihedral angle, and (iv) maximization of the minimum dihedral angle. Transformations are also used to enforce a specific connectivity into the grid without taking into account quality aspects. Such an example is the boundary recovery algorithm (section 4.6). Most types of transformations require orientation checks to ensure the geometric validity of the new elements. Robust orientation checks in this study are guaranteed by using exact geometric predicates [174].

4.5.1 ELEMENTARY FLIPS

Given five noncoplanar points in $\mathbb{R}^3$, there are four elementary flips [160]: 1-4, 4-1, 2-3, and 3-2 (Figure 51). The first number indicates the number of tetrahedra before the flip and the second number indicates the number of tetrahedra after the flip. Flips 1-4 and 4-1 are used to insert a point into the grid and to remove a point from the grid, respectively (Figure 51(a)). Flips 2-3 and 3-2 are used to reconnect five noncoplanar points. A 2-3 flip removes a face from the grid, while a 3-2 flip removes an edge (Figure 51(b)). Flips 2-3, and 3-2 are special cases in Lawson's theorem [114] for 3-dimensions. Lawson's theorem states that no more than two connectivities are possible for $k+2$ points in k dimensions.



(a) 1-4, 4-1 flip (b) 2-3, 3-2 flip

Figure 51: Elementary flips in 3-dimensions. A 1-4 flip inserts a point (p) into the grid subdividing a tetrahedron into four tetrahedrons. The inverse operation removes a point (p) from the grid. 1-4, and 4-1 flip are performed only if p is inside the tetrahedron. A 2-3 flip removes a face (abc) from the grid, creating a new edge (de). Flip 2-3 is geometrically valid if the edge de intersects face abc in the interior. The inverse operation removes an edge (de) from the grid, creating a new face (abc).

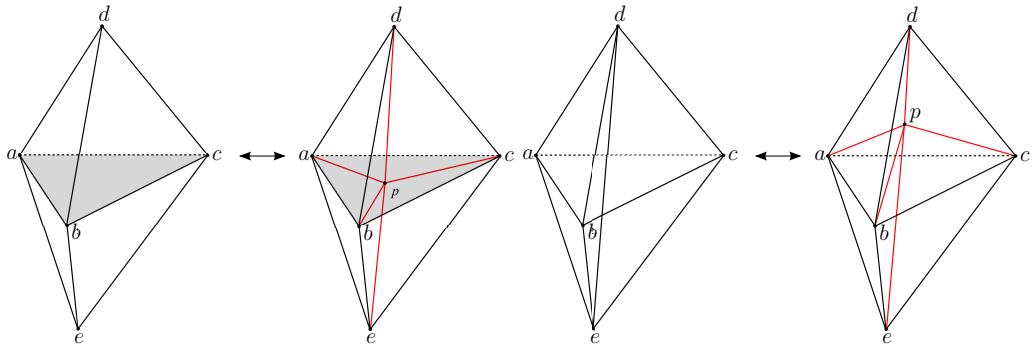
4.5.2 OTHER FLIPS

CDT3D implements a 2-6 flip and a n-2n flip (n is the number of tetrahedra surrounding an edge). A 2-6 flip inserts a point on a face (Figure 52(a)). A n-2n flip inserts a point on an edge (Figure 52(b)). The inverse operations remove a point from a face or an edge (Figure 52). A 2-2 flip reconnects five points, four of which are coplanar points (Figure 52(c)).

4.5.3 COMBINATIONS OF FLIPS

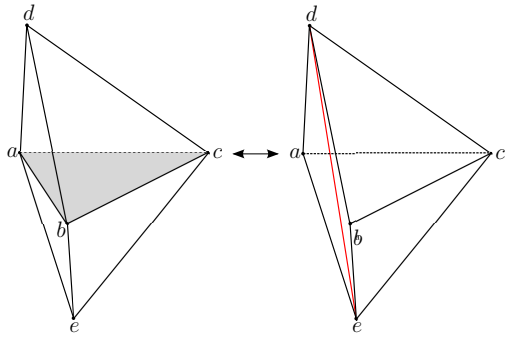
A 4-4 flip interchanges two edges in a set of four tetrahedra surrounding an edge (Figure 53). It is essentially a combination of a 2-3 flip (that inserts a new edge) and a 3-2 flip (that removes an old edge). The first 2-3 flip may temporarily create a flat tetrahedron which will be removed by the followed 3-2 flip. Two candidate configurations are computed in advance for a 4-4 flip (Figure 53). The configuration that optimizes an objective function is then selected. The 4-4 flip is implemented both as an independent operation and within a general n-m flip [178, 77, 78].

The n-m flip ($n \geq 3$, $m = 2n - 4$) is essentially a combination of $(n - 3)$ 2-3 flips followed by a final 3-2 flip, where n, m is the number of tetrahedra before and after the flip, respectively. The n-m flip has proven very effective for edge removal. However,



(a) 2-6, 6-2 flip

(b) n-2n, 2n-n flip (n = 3)



(c) 2-2 flip

Figure 52: Other flips in 3-dimensions. 2-6 flip inserts a point (p) on a face (abc), subdividing the face into three sub-faces. n-2n flip inserts a point (p) on an edge (ed), subdividing the edge into two sub-edges. 6-2 flip removes a point (p) from a face (abc), recovering the face. 2n-n flip removes a point (p) from an edge (ed), recovering the edge. 2-2 flip recovers an edge (ed) when four points (a, b, e, d) are coplanar and located on the boundary.

n-m flips can significantly deteriorate the performance of the grid generation. Indeed, [77] shows that the number of candidate tetrahedralizations for n tetrahedra surrounding an edge increases exponentially in n, thus the cost to exhibit these tetrahedralizations is a priori non-polynomial. Specifically, the number of candidate solutions is given by the Catalan number (Figure 54):

$$C_n = \frac{(2n-2)!}{n(n-1)!} \tag{7}$$

with $(n \geq 3)$. Table 15 gives the number of candidate solutions as a function of n.

Figure 55 illustrates the possible tetrahedralizations for $n = 5$ elements. Figure 56 illustrates the transformations required to obtain a tetrahedralization for $n = 5$, $m = 6$. It is not always possible to remove an edge using a n-m flip, because a face incident on this edge might not be flippable (Figure 57). To alleviate this problem, a recursive n-m flip is implemented [179]. In the recursive n-m flip, if a face incident on edge ab cannot be

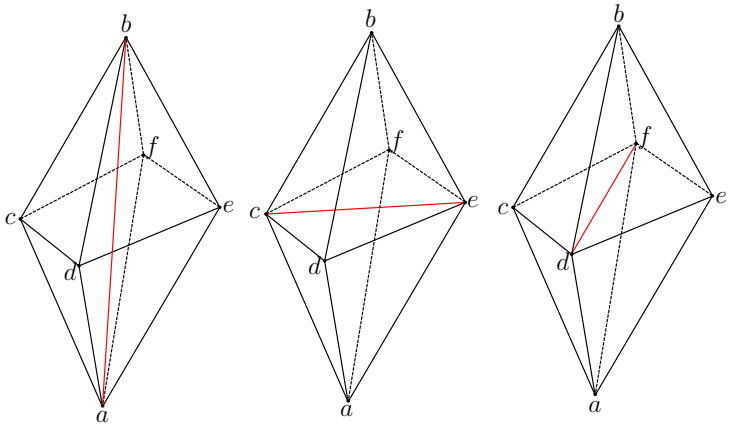


Figure 53: 4-4 flip. Left: Initial configuration with four tetrahedra ($abcd$, $abde$, $abef$, $abfc$) surrounding an edge (ab). Middle: first alternative configuration with edge ab being replaced by edge ce. The new tetrahedra are: $ceda$, $cfea$, $cdeb$, $cefb$. Right: second alternative configuration with edge ab being replaced by edge df. The new tetrahedra are: $dcfa$, $dfea$, $dfcb$, $defb$.

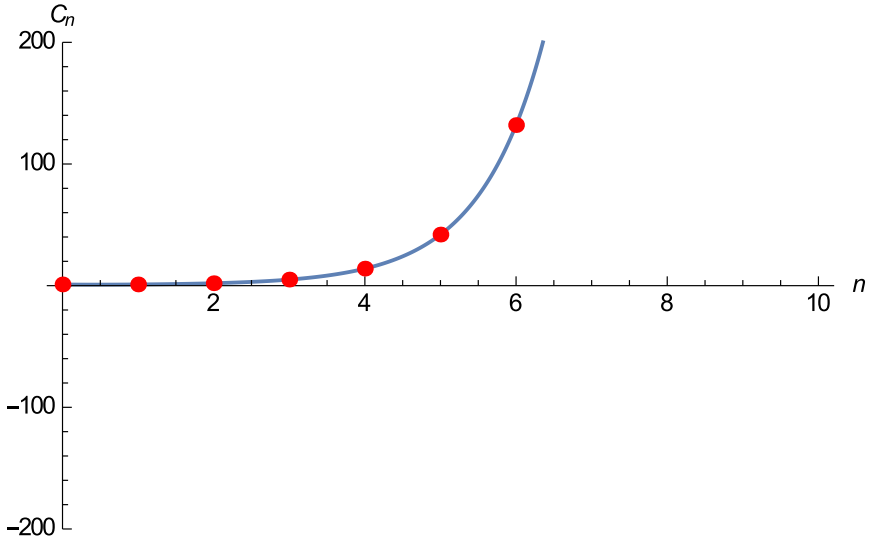


Figure 54: The Catalan number C_n as a function of n.

Table 15: Number of candidate tetrahedralizations C_n for n tetrahedra surrounding an edge. m is the number of tetrahedra after reconnection.

n	3	4	5	6	7	8	9	10
$m = 2n - 4$	2	4	6	8	10	12	14	16
C_n	1	2	5	14	42	132	429	1430

flipped, then the algorithm tries to remove another edge that contains one of the endpoints of ab (Figure 57). If the other edge is flipped, then the algorithm continues to reduce the cardinality of ab. Otherwise ab cannot be flipped.

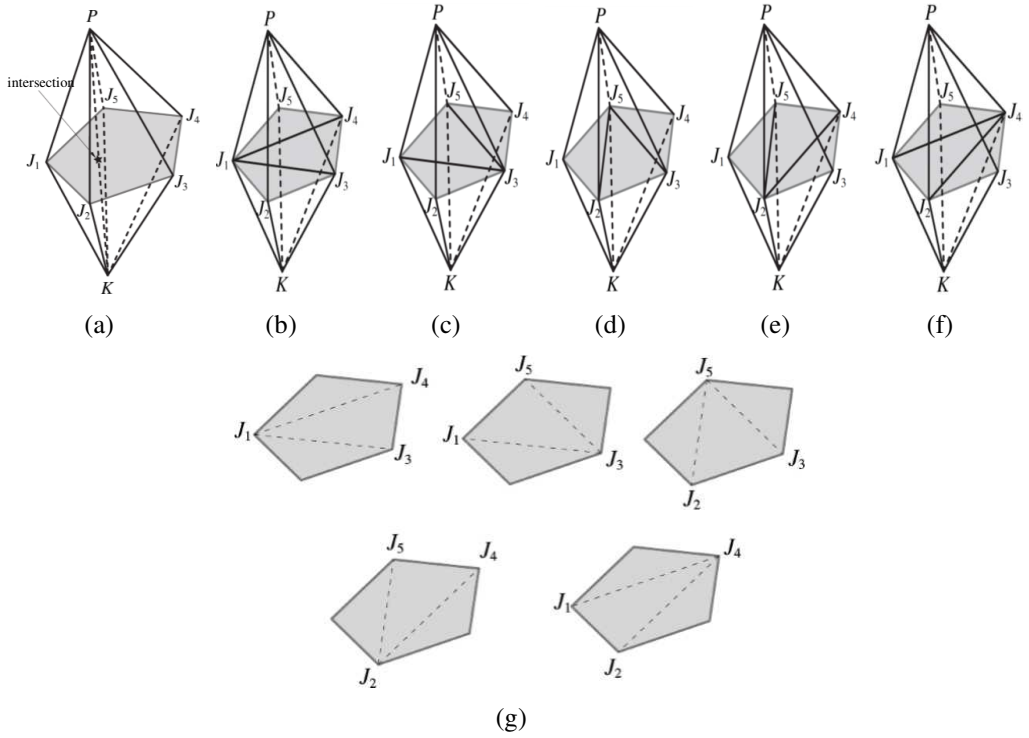


Figure 55: Possible tetrahedralizations for $n = 5$ elements surrounding edge KP. (g) depicts the possible triangulations of the polygon $J_1J_2J_3J_4J_5$.

This work adopts recursive n-m flips ($n \leq 10$) to recover the boundary domain. For performance reasons, the refinement and improvement modules reconnect clusters of up to $n = 4$ tetrahedra. When $n = 4$, the two candidate configurations are computed in advance and the one that maximizes the objective function is selected. In future work, higher order flips $(n > 4)$ will be incorporated, and their impact on the performance and the quality of the procedure will be investigated.

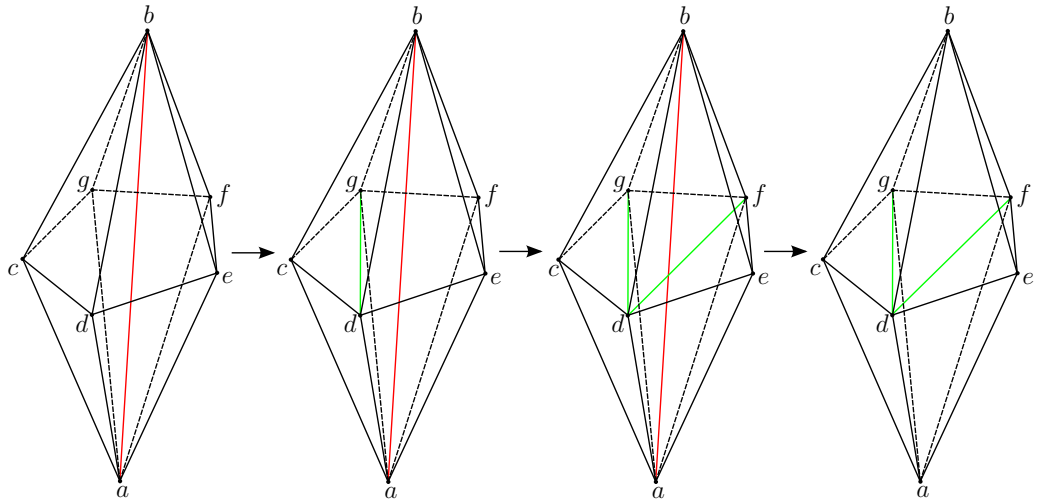


Figure 56: n-m flip ($n = 5$, $m = 6$). This example illustrates a sequence of 2-3 flips and a final 3-2 flip to reconnect five tetrahedra surrounding an edge (ab). The result tetrahedralization is one out of five possible tetrahedralizations. From left to right: Initial configuration with five tetrahedra ($abcd$, $abde$, $abef$, $abfg$, $abgc$) surrounding an edge (ab); connectivity after a 2-3 flip between $abcd$, $acbg$ (face abc is removed and edge dg is created); connectivity after a 2-3 flip between $aebd$, $abef$ (face abe is removed and edge df is created); final connectivity after a 3-2 flip between $abdf$, $abfg$, $abgd$ (edge ab is removed).

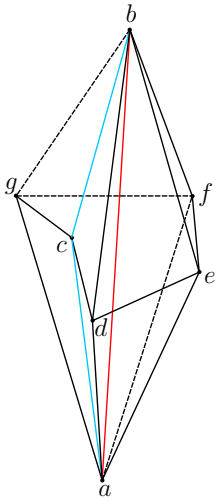


Figure 57: Non-flippable edge ab surrounded by $n = 5$ tetrahedra. Face abc cannot be removed because a 2-3 flip generates intersected tetrahedra. A recursive n-m flip attempts to remove either edge ac or bc, thus increasing the chances to flip ab.

4.6 BOUNDARY RECOVERY

Many applications require the preservation of a set of constraints (i.e., edges and faces) into the grid. The constraints can represent boundaries of the domain, an interface of a mesh partition, or other geometric features. Unfortunately, in $\mathbb{R}^3$, given a set of constrained points and faces and a corresponding triangulation obtained from a Delaunay tetrahedralization of those points, there is no guarantee that the constrained edges or faces will be contained within the Delaunay grid. Additional (*Steiner*) points may be necessary in order to recover the constraints. The optimal locations and (minimum) number of Steiner points is still an open problem. In general, proving the existence of a tetrahedral grid of an arbitrary polyhedron is reported to have a non-polynomial complexity [165, 31]. Two popular examples of polyhedra with no tetrahedralization without Steiner points are the Schönhardt polyhedron [170] and the Chazelle's polyhedron [30]. All the above facts make the boundary recovery a challenging problem.

This thesis implements a boundary recovery algorithm which is inspired by state-of-the-art software [78, 196, 179]. Our work does not provide any theoretical guarantees for the minimum number of Steiner points. However, our boundary recovery algorithm exhibits "ultimate" robustness; it is capable to recover closed manifold surfaces with extremely low quality or high aspect-ratio triangles, by using only few Steiner points or without any Steiner points (Table 16).

The boundary recovery procedure includes three steps:

1. Recovery of the constrained edges (by means of edge partitioning).
2. Recovery of the constrained faces (by means of face partitioning).
3. Partition elimination.

4.6.1 RECOVERY OF THE CONSTRAINED EDGES AND FACES

This step first recovers the constrained edges and then the constrained faces. The missing constrained entities (edges/faces) are recovered in two sub-steps. In the first sub-step, the edges/faces intersecting a constrained entity are identified. For this purpose, robust intersection checks between mesh entities (i.e., edge-to-edge, face-to-face, and edge-to-face) are implemented based on exact geometric predicates [174].

In the first sub-step, the algorithm attempts to remove as many intersecting edges/faces as possible by using recursive n-m flips (Figure 56). In the second sub-step, the constrained

entities that could not be recovered from the first sub-step are subdivided. A constrained edge is subdivided with a n-2n flip (Figure 52(b)). A constrained face is subdivided with a 2-6 flip (Figure 52(a)).

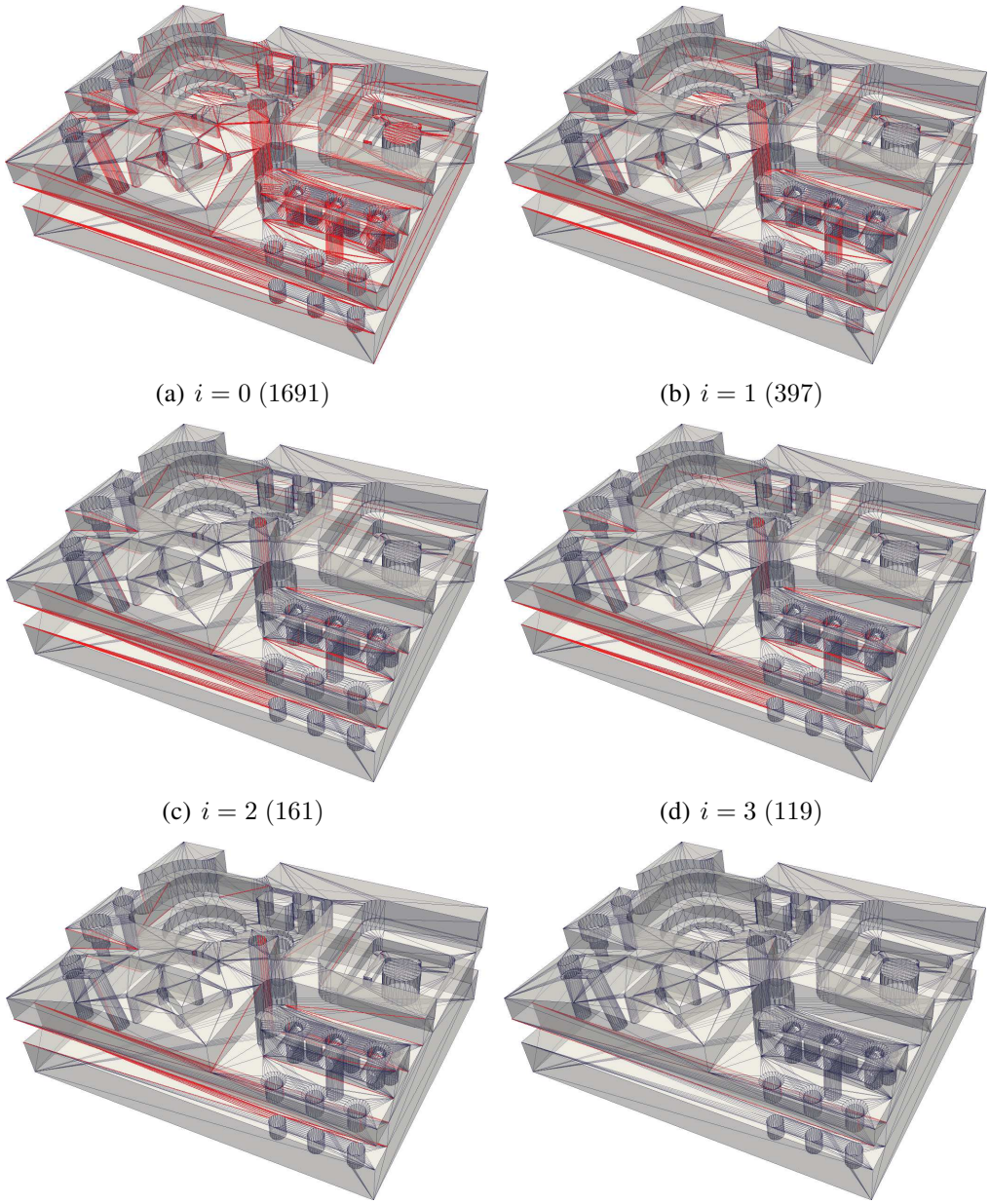
The unrecovered edge or face partitions are then used as an input for the next iteration, and the two sub-steps repeat until all constraints or their partitions are recovered. After the completion of this step the tetrahedra located outside the recovered (partitioned or not) boundaries are discarded.

This study balances the running time for recursive n-m flips and the number of added points to enforce partitioning by limiting the depth of the recursion and by checking configurations with a limited n (number of tetrahedra surrounding an edge). Experimental evaluation showed that when depth $= 5$ and $n = 10$ then only few Steiner points are inserted on the boundary within a reasonable amount of time (depending on the size and the quality of the boundary triangulation). The algorithm can also adjust progressively the depth and n based on the number of missing entities in each iteration. Figure 58 depicts an example during edge recovery. Figure 59 depicts an example after edge recovery and face recovery is completed. Edge partitioning does not guarantee a valid initial face partitioning. For this reason, the initial sub-faces are constructed before the face recovery. A 2-dimensional triangulator for convex polygons is implemented to construct those sub-faces (Figure 60).

4.6.2 PARTITION ELIMINATION

The edge/face partitions are removed from the boundary with the following operations:

1. For each Steiner point on the boundary:
   (a) Identify the polygon formed by all the boundary faces surrounding the point.
   (b) Identify the set of tetrahedra surrounding the point (ball).
   (c) Remove the point and the tetrahedra in the ball.
   (d) Triangulate the polygon to enforce the initial boundary edges/faces or their partitions.
   (e) Tetrahedralize the empty ball by inserting a new Steiner point inside the ball.
2. For each Steiner point in the volume:
   (a) Collect all the edges surrounding the point.



(a) $i = 0$ (1691)

(b) $i = 1$ (397)

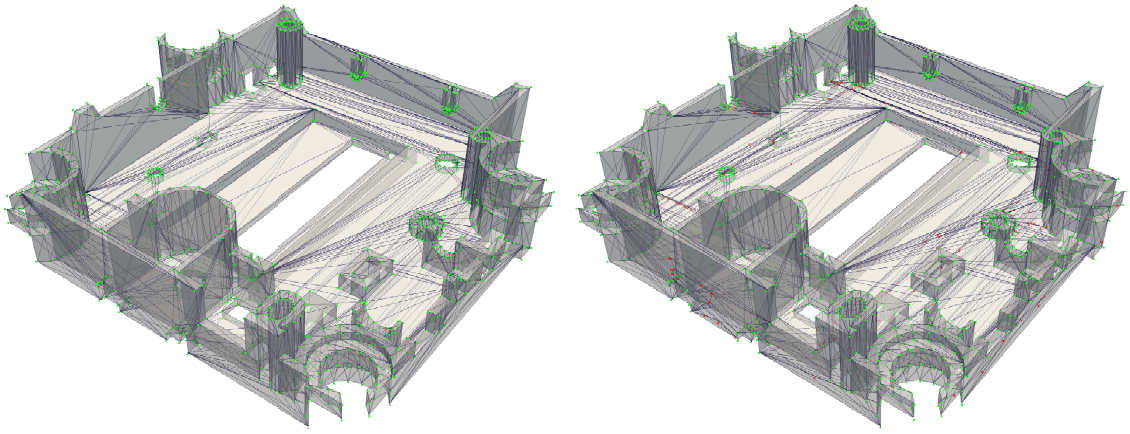
(c) $i = 2$ (161)

(d) $i = 3$ (119)

(e) $i = 10$ (75)

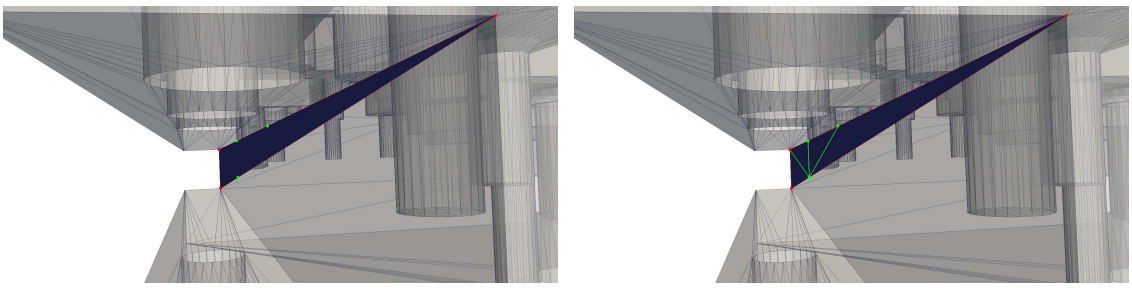
(f) $i = 14$ (9)

Figure 58: Edge recovery with 14 iterations. (a) depicts the missing edges (red) before edge recovery. (b)-(f) depict the missing edges during edge recovery. The number in the parenthesis denotes the number of missing edges at iteration i. The input triangulation (mohne) obtained from INRIA's mesh database (https://www.rocq.inria.fr/gamma/gamma/download/download.php).



(a) (b)

Figure 59: Comparison between the input constrained triangulation (a), and the recovered triangulation after edge/face partitioning (b). Number of input constrained points (green): 1261. Number of additional Steiner points (red): 181. The Steiner points will be suppressed in a subsequent partition elimination step. The input triangulation (Mech-shell) obtained from INRIA's mesh database (https://www.rocq.inria.fr/gamma/gamma/download/download.php).



(a) (b)

Figure 60: Construction of an initial face partition for face recovery. (a) depicts a polygon formed by edge partitioning. The polygon has 3 constrained vertices (red) and 3 Steiner vertices (green). (b) depicts a valid triangulation of the polygon with 4 sub-faces.

(b) Try to remove each edge with a n-m flip.

(c) If four edges remain after collapsing then use a 4-1 flip to remove the point.

Figure 61 depicts a ball. Figure 62 depicts an example with partition elimination. Special care is required when a Steiner point is inserted on an edge and the edge is common to two initial faces not in the same plane. In this case, two sub-polygons are re-triangulated, one for each plane. Figure 62(a) depicts such a case. This work assumes a manifold surface triangulation, so the maximum number of planes is two.

After the completion of the partition elimination, all the introduced Steiner points are relocated inside the volume and the boundary triangulation is identical with the input boundary triangulation. Most relocated Steiner points can be removed after collapsing the edges surrounding a point. In practice, only a few Steiner points may remain inside the volume.

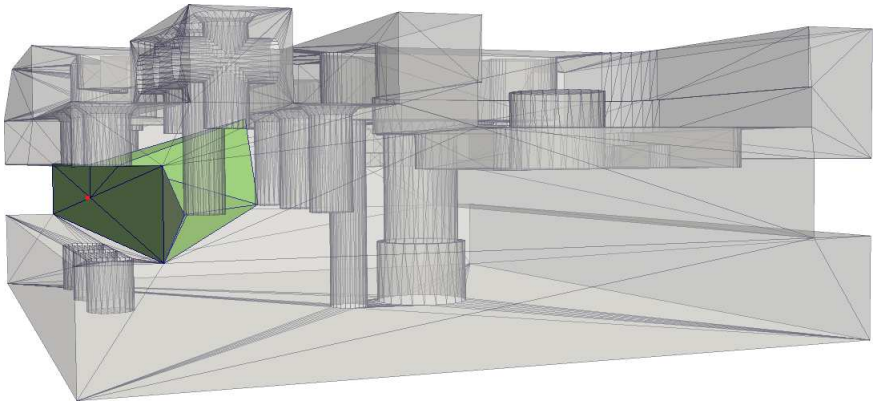


Figure 61: A ball of tetrahedra (green) surrounding a Steiner point (red) on the boundary. The point will be relocated inside the ball and the ball will be re-tetrahedralized. The original faces will be enforced on the boundary.

4.6.3 EXAMPLES WITH BOUNDARY RECOVERY

The boundary recovery algorithm is evaluated on six challenging manifold triangulations (Figure 63). Triangulations (a)-(d) simulate badly discretized domains as they contain a high number of triangles with large and small angles. Triangulations (e)-(f) simulate surfaces that intersect a boundary layer region as they contain high-aspect ratio triangles. Figure 64 depicts a closer view and quality statistics for two input triangulations. In all examples, a Delaunay tetrahedral grid is initially generated by inserting the points of the triangulation. The boundary is then recovered.

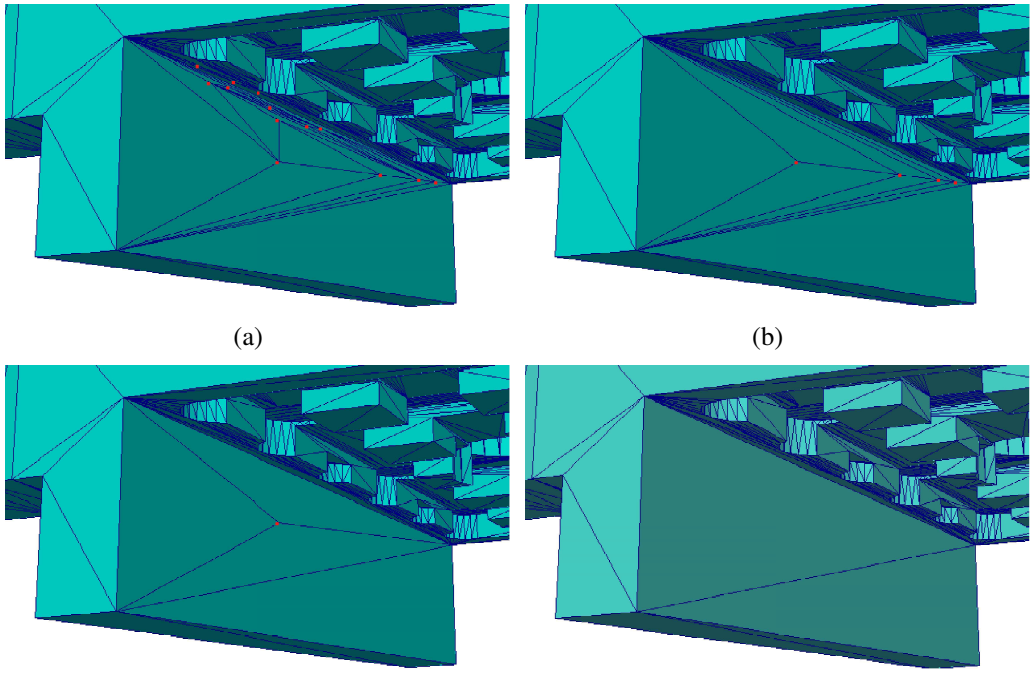
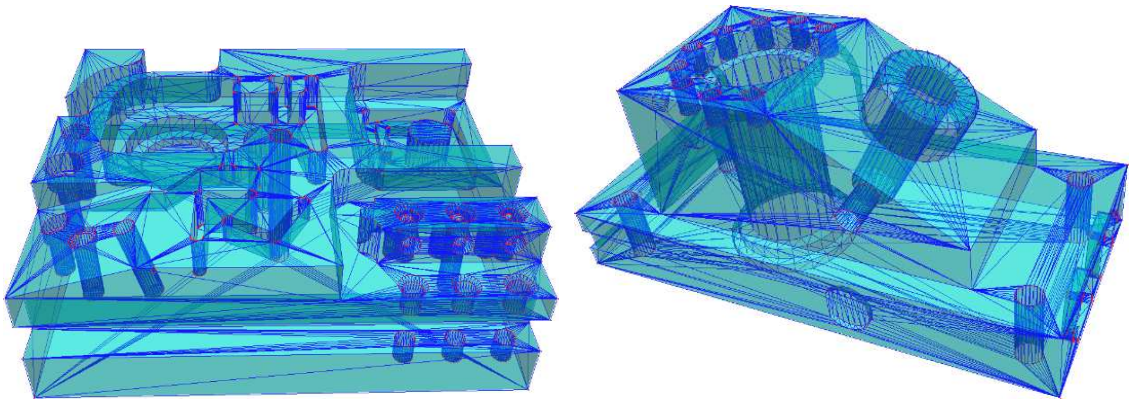
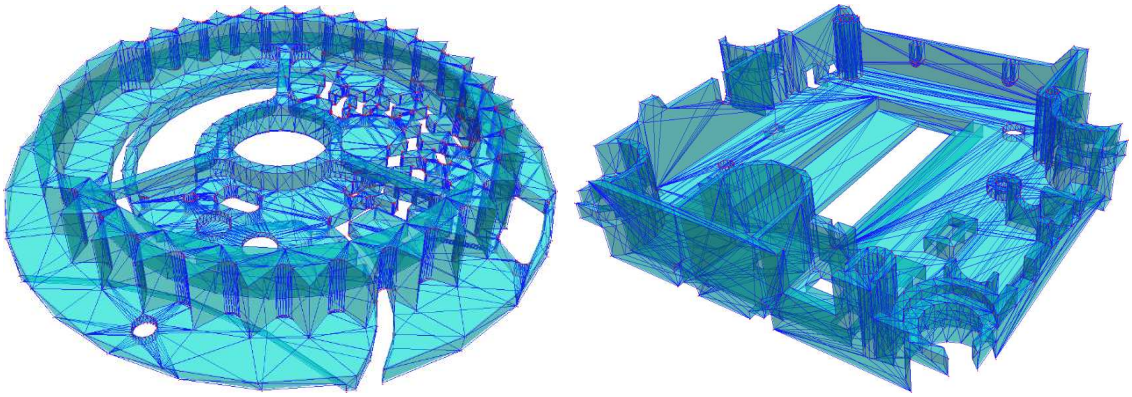


Figure 62: Partition elimination procedure. (a) depicts the boundary edge/face partitions before elimination. Red reperesents the added Steiner points. (b)-(c) depict the boundary edge/face partitions during elimination. (d) depict the recovered boundary after elimination.

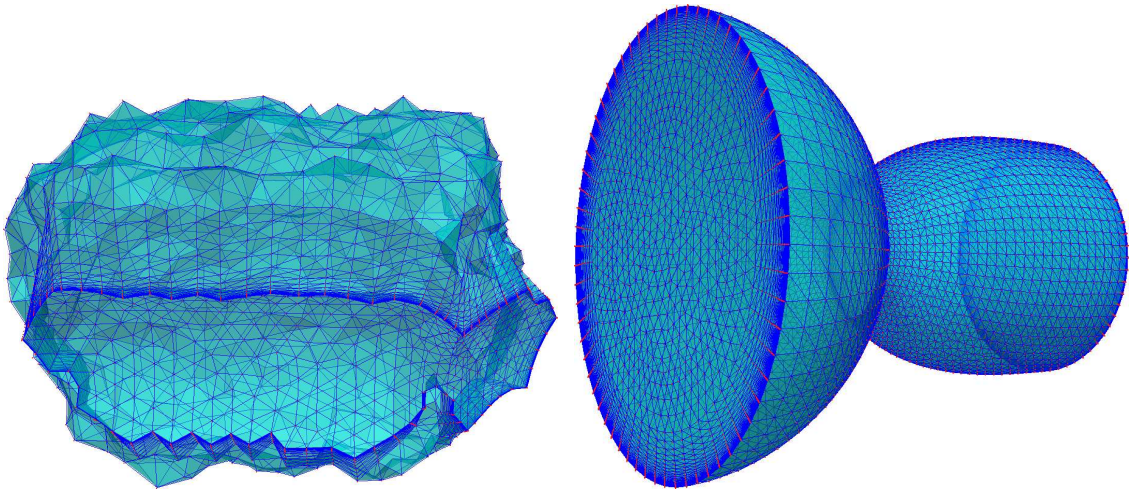
Table 16 reports the results. CDT3D exhibits "ultimate" robustness on recovering the boundary triangulation. In most of the cases, it successfully recovers the boundary only with only a few Steiner points inserted in the volume. At Mech-shell case, TetGen fails to remove one Steiner point from the boundary. At Submesh1 case, TetGen terminates the program with segmentation fault. In practice, the input surface contains triangles with better shape than in those six examples, although layered anisotropic triangles or few triangles with small and large planar angles might still exist. In all those configurations, our implemented algorithm recovers the boundary surface either without any additional points or with only a few (volume) points.



(a) Mohne (b) Anc101

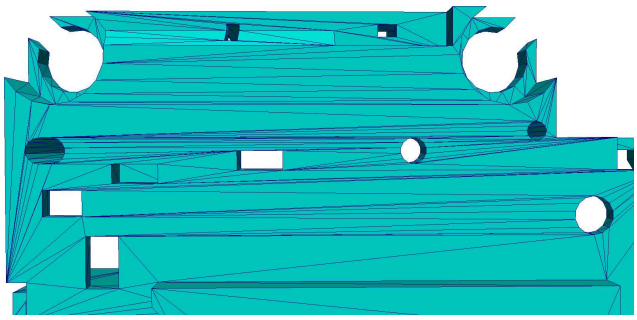


(c) Thru-maze (d) Mech-shell

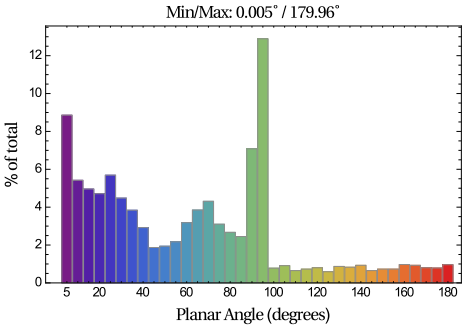


(e) Submesh1 (f) Nozzle

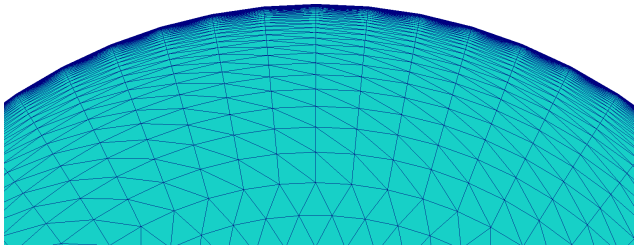
Figure 63: Triangulations for boundary recovery. (a)-(d) obtained from INRIA's mesh database (https://www.rocq.inria.fr/gamma/gamma/download/download.php). (e)-(f) obtained from the CAVS Sims Center at MSU.



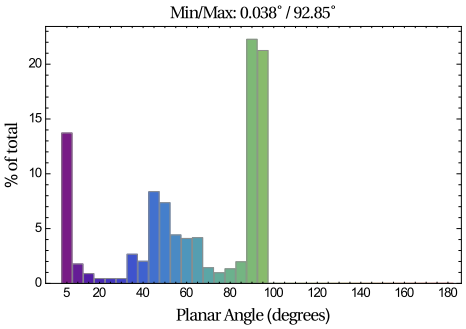
(a) Badly shaped triangles (Mech-shell)



(b) Quality histogram (Mech-shell)



(c) Layered anistropic triangles (Nozzle)



(d) Quality histogram (Nozzle)

Figure 64: Detail views and planar angle histograms (in 5-deg increments) of Mech-shell and Nozzle triangulations. The planar angle extrema are reported in each histogram.

Table 16: Evaluation results on boundary recovery. CDT3D is compared with state-of-the-art open-source technology TetGen v1.5.0 [179]. #Stpts is the number of Steiner points. The experiments performed on a Linux desktop machine with Intel Core i7-2600 CPU@3.40GHz, and 16 GB RAM.

Triangulation	Mohne		Anc101		Thru-maze		Mech-shell		Submesh1		Nozzle	
#points	2760		1378		2781		1261		3860		9644	
#edges	8340		4158		8433		3921		11574		28926	
#faces	5560		2772		5622		2614		7716		19284	
Software	TetGen	CDT3D	TetGen	CDT3D	TetGen	CDT3D	TetGen	CDT3D	TetGen	CDT3D	TetGen	CDT3D
#Stpts (boundary)	0	0	0	0	0	0	1	0	-	0	0	0
#Stpts (volume)	7	2	0	0	8	5	59	28	-	0	0	0
Time (sec)	0.18	0.29	0.08	0.11	0.11	0.09	0.18	2.6	-	0.36	0.02	0.20

4.7 REFINEMENT

The proposed refinement algorithm is essentially a combination of Advancing Front type point placement, direct element subdivision, and parallel multi-threaded connectivity optimization schemes. CDT3D is inspired from state-of-the-art unstructured grid technology AFLR (Advancing Front-Local Reconnection) [132, 131]. AFLR is directly incorporated in several systems, including: DoD CREATE-MG Capstone, Lockheed Martin/DoD ACAD, Boeing MADCAP, MSU SolidMesh, and Altair HyperMesh.

The contributions of the proposed refinement scheme are:

1. Rapid local reconnection using parallel multi-theaded topological transformations. Up to 2.5 and 10 times faster refinement compared to state-of-the-art software [132], with 1 and 12 hardware cores, respectively.

2. Element quality comparable with state-of-the-art code [132]. Very low number of slivers (typically $< 0.005\%$, Table 18).

Details of the refinement procedure are presented in the following subsection.

4.7.1 PROCEDURE

During the refinement, new points are introduced into the grid until the grid satisfies a desired point distribution function. New points are created with an Advancing Front type point placement. Points are inserted by directly subdividing the tetrahedra that contain them. Finally, the grid is reconnected in parallel with a combination of topological transformations. The above operations are repeated until a complete field grid is generated with a desired point distribution. Two objective functions are used to optimize the connectivity: (i) Delaunay empty sphere criterion [51], and (ii) Maximization of the minimum Laplacian edge weight [14].

Given a tetrahedral grid that contains all the surface edges/faces, the basic steps of the procedure are outlined as follows:

1. Compute distribution function for each point on the boundary.

2. Make all tetrahedra active (i.e., eligible for reconnection).

3. While new field points are accepted:

   (a) Deactivate tetrahedra that satisfy the point distribution function.

(b) Create new field points.

(c) Insert field points.

(d) Optimize connectivity.

The details of each step are presented in the following paragraphs.

Initialize distribution function on the boundary

The average of the boundary edge lengths surrounding a point is used for the initial distribution function. The point distribution is used to control the field point spacing. The final field point spacing is compatible with the boundary and varies smoothly between boundaries. The growth of field spacing is controlled with the distribution function weighting factor $cdfm \in [0, 1]$. A value $cdfm > 0$ reduces the growth of element size from small to larger elements and increases the total number of grid points generated. Examples of grids generated for varied $cdfm$ are presented in the next section.

Element activation

A set of faces, called front, is maintained throughout the grid generation process. The front may contain boundary faces of active tetrahedra and faces between active and inactive tetrahedra. Initially all tetrahedra are marked as active, hence the front is the boundary triangulation.

Element deactivation

An element becomes inactive if: (i) the length of each edge is less than $cdff \in [1.1, 3]$ times the average distribution function of the edge points, and (ii) its maximum dihedral angle is less than $angmax \in [120, 179.99]$. The default values are $cdff = 1.5$ and $angmax = 150$.

Point creation

New candidate field points are created by advancing from selected faces of the front (Figure 65). A new candidate point is created by advancing in a direction normal to the selected face a distance that would produce an equilateral element based on an appropriate length scale. The average of the point distribution function on the face points is used as a length scale. To improve the grid quality, the length scale is limited to be less than

$cdfr \in [1, 3]$ times the average length of the face edges. The default value is $cdfr = 1.1$. Candidate points advancing from two selected faces of the front can be averaged to improve element quality in regions near boundary discontinuities. A new candidate point is rejected if at least one criterion from the following is true:

1. The candidate point is outside the boundary or on the boundary.
2. The candidate point is too close to the boundary.
3. The candidate point is too close to the containing tetrahedron points.
4. The candidate point is too close to an other candidate point in the containing tetrahedron or an other candidate point in the tetrahedra immediately attached to the containing tetrahedron.

CDT3D implements a stochastic walk algorithm [52] with robust geometric predicates [174] to locate the candidate point. If the candidate point is located on a face, then the containing tetrahedron is the set of two tetrahedra sharing this face. If the candidate point is located on an edge, then the containing tetrahedron is the set of all tetrahedra sharing this edge. Once the containing element is found, the point distribution function is linearly interpolated from the distribution function at the containing tetrahedron points.

The distance between candidate points or between a candidate point and an existing point is not allowed to be less than $cdfn \in (0.1, 2]$ times the larger of the distribution function for the candidate point or for the other point being checked. The default value is $cdfn = 0.7$ The distance between a candidate point and its projection to the closest boundary face is not allowed to be less than $cdfn$ times the larger of the distribution function for the candidate point or for the average distribution function of the points on the boundary face being checked. The grid refinement completes when no new candidate field points are created or accepted. Figure 66 depicts an example of accepted candidate points at the first refinement pass.

Point insertion

Points are inserted into the grid by directly subdividing the tetrahedra that contain them (Figure 67). Points located inside a tetrahedron are inserted into the grid using a 1-4 flip (Figure 51(a)). Points located on an interior face are inserted into the grid using a 2-6 flip (Figure 52(a)). Points located on an interior edge are inserted into the grid using a n-2n flip (Figure 52(b)).

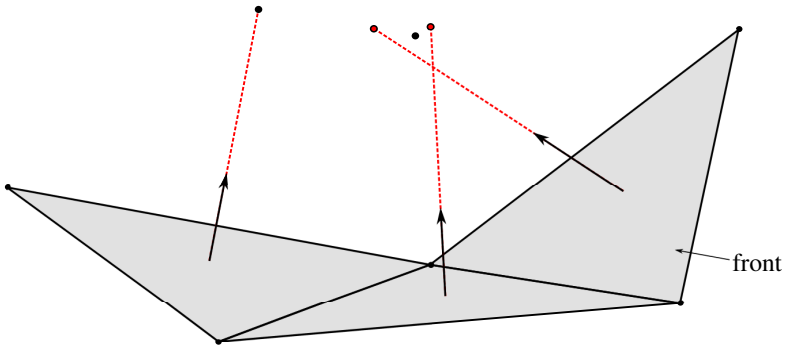


Figure 65: Advancing Front type point placement. Candidate points advancing from two selected faces of the front can be averaged to improve element quality.

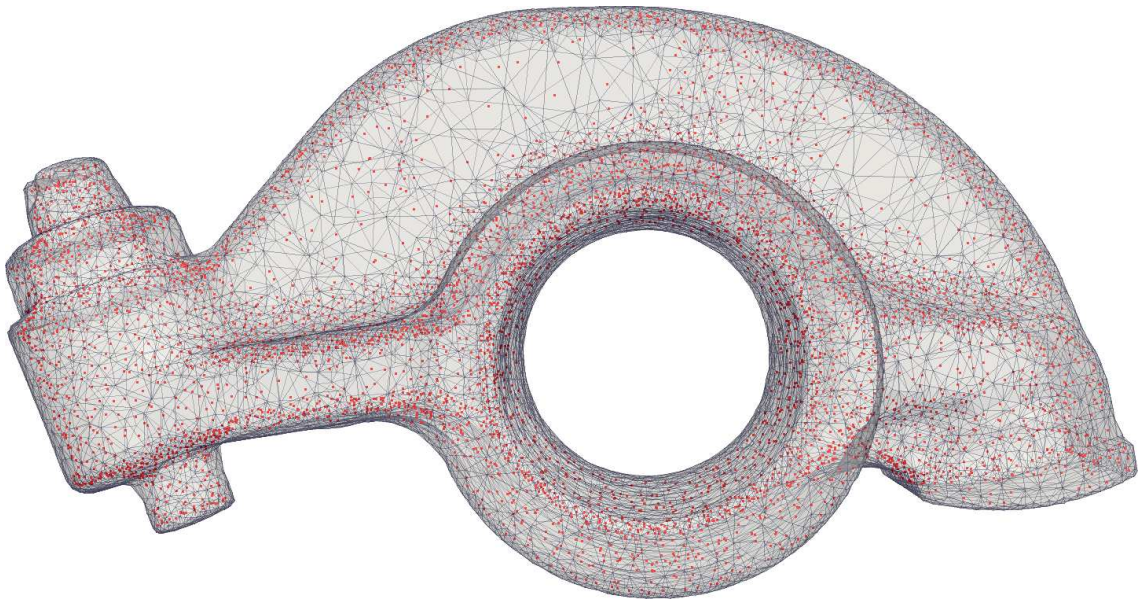


Figure 66: Accepted candidate points (red) generated at the first refinement pass with an Advancing Front type point placement.

After each insertion, the containing element is subdivided, hence the next candidate point needs to locate again. Finding a suitable tetrahedron to start the search for point location can be time-consuming. Similar to the Delaunay point insertion (subsection 4.4), locality is guaranteed during the insertion of the field points, and the running time is significantly reduced. Specifically, each tetrahedron stores a list of all the accepted candidate points contained in it. The points in each list are inserted consecutively into the grid, and the most recently created tetrahedron is selected to be used as the starting element for the next point location. If during insertion a point is located on a face or on an edge, then two or more lists are merged, respectively (Figure 67).

This insertion process is suitable for parallelization. The list of points can be inserted concurrently without synchronization between threads given that lists belonging to adjacent tetrahedra do not contain points located on a face or an edge of the grid.

If the inserted point is located too close to an interior edge or an interior face, then the point may be inserted as it was located exactly on the edge or the face, respectively, in order to improve the quality of the new elements.

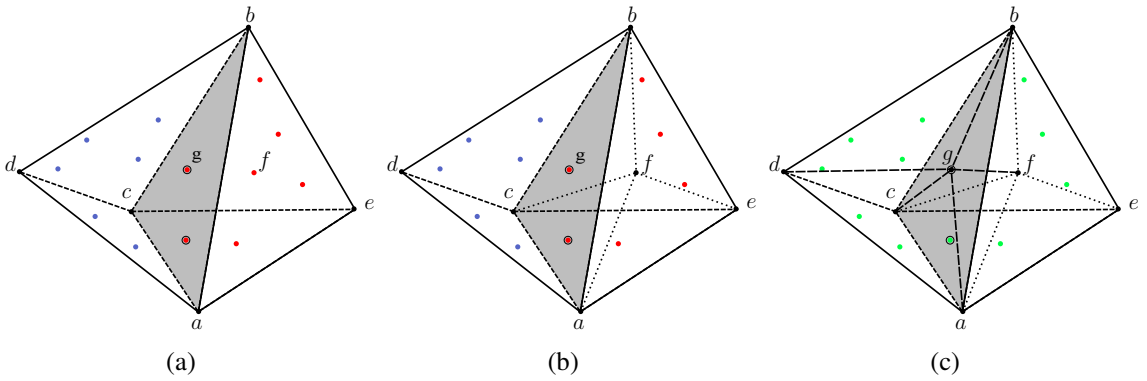
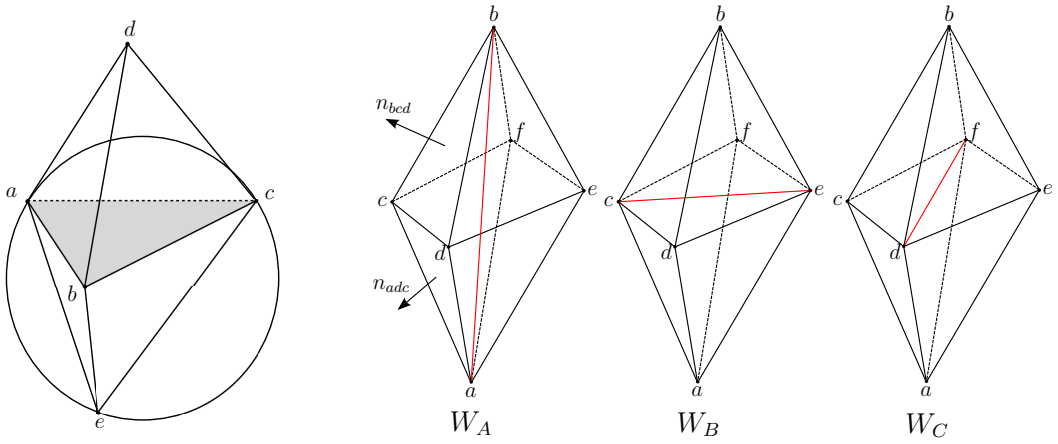


Figure 67: Direct point insertion. $acbd$ contains the blue generated points, and $abce$ contains the red generated points (two red points are located on face abc). (a) depicts the grid connectivity before the points are inserted. (b) depicts the grid connectivity after inserting point f using a 1-4 flip ($abce$ is replaced by $afcb$, $efab$, $cfeb$, $aecf$). The last created tetrahedron $(aecf)$ is used as start tetrahedron to locate the next inserted point. (c) depicts the grid connectivity after inserting point g. g is located on a face thus a 2-6 flip is used ($abcd$, and $afcb$ are replaced by six new tetrahedra). After inserting g, the blue and the red point sets are merged (green).

Connectivity optimization

Obviously the connectivity created with direct element subdivision is not acceptable. Our procedure uses combinations of local topological transformations to improve the grid connectivity. The topological transformations are applied repetitively in order to produce a global effect. Two quality criteria are used to improve the connectivity: (i) Delaunay type (Figure 68(a)) [51] and (ii) Min-Max type (maximization of the minimum Laplacian edge weight, Figure 68(b)) [14].

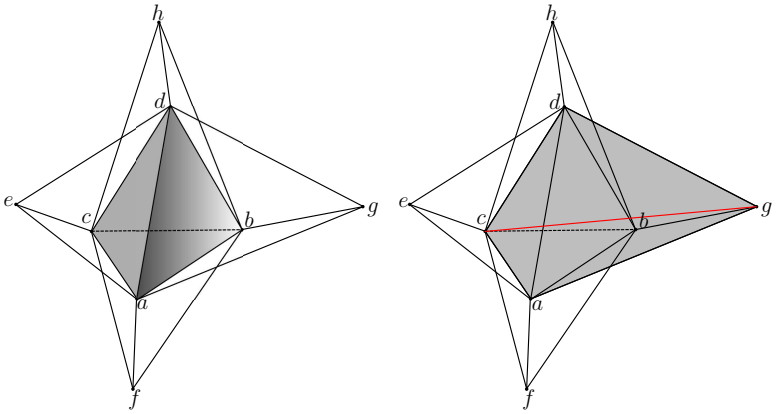


(a) In-sphere test (b) Maximization of the minimum Laplacian edge weight

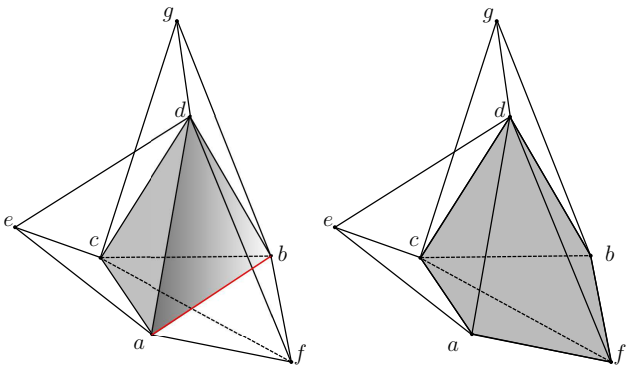
Figure 68: Quality criteria for reconnection. The grid is reconnected with a Delaunay type criterion (a) followed by a Min-Max type criterion (b). In (a), five points are reconnected (using a 2-3 flip) only if d is inside the circumsphere of $acbe$. In (b), six points are reconnected with a configuration that maximizes a Laplacian edge weight $W = \max(W_A, W_B, W_C)$. The laplacian weight is obtained for a given tetrahedron edge from the negative value of the inner product of the two area vectors common to the opposite edge divided by the volume [14]. For edge ab is: $W_A = W_{ab} = -\left(\frac{n_{bcd}\cdot n_{adc}}{V_{abcd}} + \frac{n_{bde}\cdot n_{aed}}{V_{abde}} + \frac{n_{bef}\cdot n_{afe}}{V_{abef}} + \frac{n_{acf}\cdot n_{bfc}}{V_{abfc}}\right)$

In each refinement iteration, the grid is optimized in two passes. In each pass, tetrahedra are reconnected until no new reconnections are possible. In the first pass the grid is reconnected with a Delaunay type criterion. In the second pass, the grid is reconnected with a Min-Max type criterion. The combined criterion eliminates most field sliver elements and produces a grid with quality superior to that of a grid generated using only a Delaunay criterion. The grid is reconnected throughout the entire generation process to obtain a maximum quality [132].

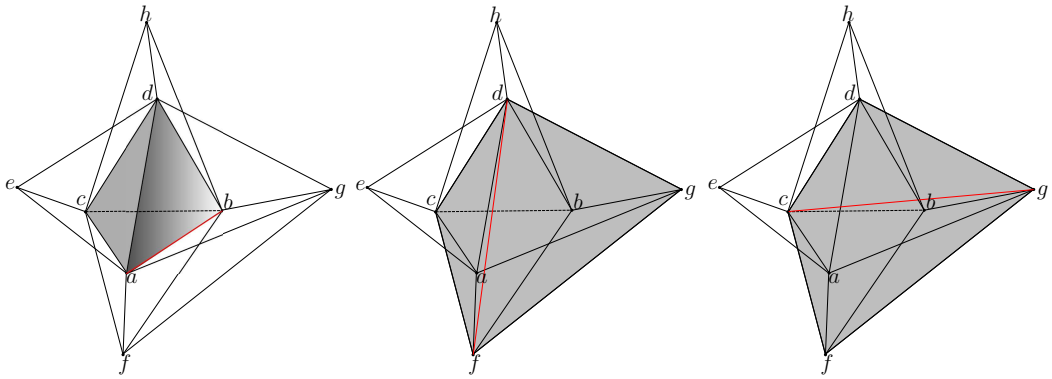
Figure 69 depicts the reconnection scheme. Active tetrahedra are eligible to reconnect with a neighbor. The neighbor can be either active or inactive. After reconnection, new



(a) 2-3 flip. Face abd is removed and edge cg is created.



(b) 3-2 flip. Edge ab is removed and face cfd is created.



(c) 4-4 flip. Edge ab is replaced by edge fd or cg.

Figure 69: Local reconnection scheme. In (a) and (b), the left figure depicts the topology before reconnection, and the right figure depicts the topology after reconnection. In (c), the left figure depicts the topology before reconnection, and the middle and right figures depict the two candidate topologies after reconnection. Initially, the active element (left with grey) is attempted to reconnect with a neighbor by using a 2-3 flip (a). If a 2-3 flip is not possible, then the tetrahedra surrounding an edge on the shared face between the active element and the neighbor are checked for reconnection. Three tetrahedra are reconnected using a 3-2 flip (b). Four tetrahedra are reconnected using a 4-4 flip (c). New tetrahedra are made active after reconnection. Red indicates an edge which is removed or a newly created edge.

tetrahedra are activated and then considered for reconnection during the next traversal of the element list.

4.8 PARALLEL CONNECTIVITY OPTIMIZATION

Local reconnection with topological transformations has proven very effective for grid optimization [132, 73, 77, 97, 178]. Nevertheless, it can be the bottleneck for the performance of the grid generation. The experimental evaluation of this study indicates that the proposed sequential reconnection scheme accounts for about 80% of the total refinement time. For this reason, this thesis implements a new parallel optimistic approach to significantly reduce the overheads. To the best of our knowledge, CDT3D is the first software that optimizes the connectivity using parallel tightly-coupled topological transformations throughout the grid generation (including a post-improvement step).

4.8.1 PARALLEL PROCEDURE

The connectivity is optimized using tightly-coupled topological transformations. The parallelization is based on:

1. Over-decomposition.
2. Thread safe operations.
3. Load balancing.

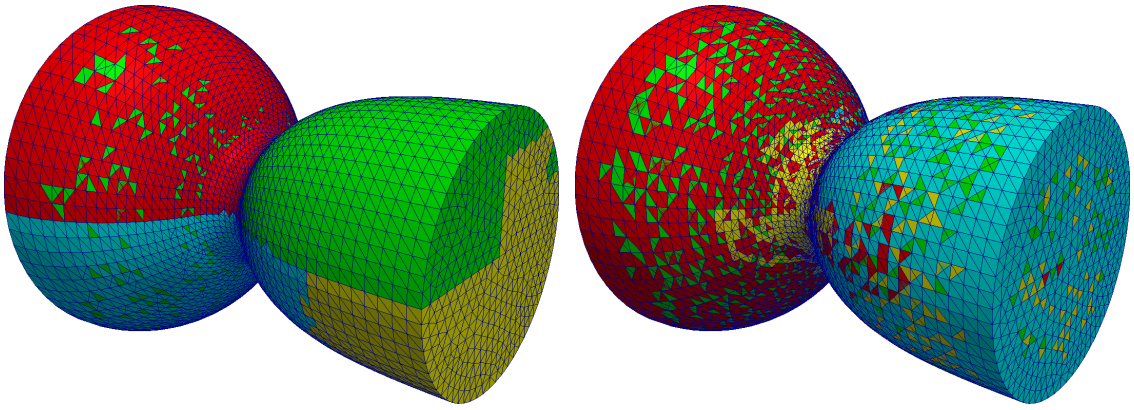
Details of the procedure are presented in the following paragraphs.

Over-decomposition

Before the reconnection, the active elements are decomposed into *work-units* (*buckets*). The granularity (grain size) of the decomposition is adjusted with parameter $nbuckets \in [nthreads, nactelem]$, where $nthreads$ and $nactelem$ are the number of threads and the number of active elements, respectively. After decomposition, each bucket contains approximately the same number of elements $(nactelem/nbuckets)$, and each thread owns approximately the same number of buckets $(nbuckets/nthreads)$.

Optionally, the active elements are pre-sorted for an initial load-balanced distibution of local operations [171]. Figure 70 depicts a decomposition of a tetrahedral grid of a nozzle with and without element pre-sorting. Element pre-sorting potentially reduces the

conflicts between polyhedra whose vertices are attempting to reconnect concurrently. Nevertheless, pre-sorting is not sufficient for balancing the work load throughout the parallel procedure because: (i) the number of geometrically valid transformations (i.e., those producing non-intersecting elements), and (ii) the number of optimum transformations (i.e., those optimizing the objective function), are not known a priori and may vary significantly among the threads. This thesis implements a customized load balancer to migrate work-units from a busy thread to threads without work (waiting threads), thereby exploiting more parallelism. Subsection 4.10.5 reports results on the performance of the parallel reconnection for varied granularities. Subsection 4.10.7 reports results on the performance of the parallel reconnection with and without element sorting.



(a) With element pre-sorting (b) Without element pre-sorting

Figure 70: Decomposition of a tetrahedral grid of a nozzle into 4 buckets with and without element pre-sorting. The centroids of the elements are sorted using a Biased Randomized Insertion Order (BRIO) [2] and a Hilbert curve [22]. Pre-sorting can potentially speedup the reconnection because it reduces the conflicts between polyhedra whose vertices are attempting to reconnect concurrently.

Thread safe operations

Each thread maintains a unique list of buckets throughout the reconnection. Each thread processes the buckets in a consecutive manner. The elements within each bucket are repetitively reconnected until no new reconnections are possible. Atomic operations are employed to avoid data races when more than one threads attempt to reconnect concurrently the same set of elements. The value of the atomic object (i.e., a vertex id) remains unchanged throughout the topological transformation (i.e., orientation test, quality test, and element creations or removals). If a reconnection is not possible because of

conflicts with other threads, the operation is cancelled and the next available neighbor of the active element is checked for reconnection. If reconnection is not possible without any conflicts (i.e., the reconnection produces invalid elements or the objective function is not optimized), then the neighbor is marked to avoid re-checking. GCC's atomic built-in functions are utilized for synchronization since they perform faster than the conventional pthread try_locks [70].

To avoid communication when elements are inserted in a bucket, the newly created elements are inserted into the bucket of the thread that performs the reconnection. The old elements are removed from a bucket only if this bucket belongs to the thread that performs the reconnection; otherwise the element is marked as invalid, and the thread that owns the bucket removes it in a later step.

Load balancing

This study implements a customized load balancer based on work-stealing [21]. The load balancer migrates work units between the threads. When a thread T_i has finished processing its own list of buckets, it pushes its id into a global list (*Waiting List*) that tracks down threads without work. Then T_i waits for other threads to advance without blocking and can be awakened by an other thread T_j right after T_j gives some work to T_i. Every time a running thread T_j completes the processing of one bucket, it checks if the *Waiting List* contains any threads. If *Waiting List* is empty, then T_j continues to the processing of the next bucket. If *Waiting List* contains threads, then T_j transfers a fraction of its unprocessed buckets to those threads. Parameter $frbtransf \in (0, 1]$ adjusts the fraction of work to be transferred. A typical value is 30% but different values can be used for more or less aggressive behaviour. Subsection 4.10.5 reports results on the performance of the parallel reconnection for varied fractions. The work to be transferred between a pair of threads T_i, and T_j can be reduced after each transfer with the coefficient $cbtransf \in (0, 1]$. The default value is 1.

The amount of work (i.e., number of buckets) to be transferred from T_i to T_j is:

$$W_{T_i \to T_j} = \left(cbtransf_{T_i \to T_j}\right)^{count_{T_i \to T_j}} \cdot frbtransf \cdot W_{T_i, free} \tag{8}$$

where $count_{T_i \to T_j}$ is the number of transfers from T_i to T_j and $W_{T_i, free}$ the unprocessed number of buckets in T_i. Figure 71 illustrates an example with load balancing.

4.9 QUALITY IMPROVEMENT

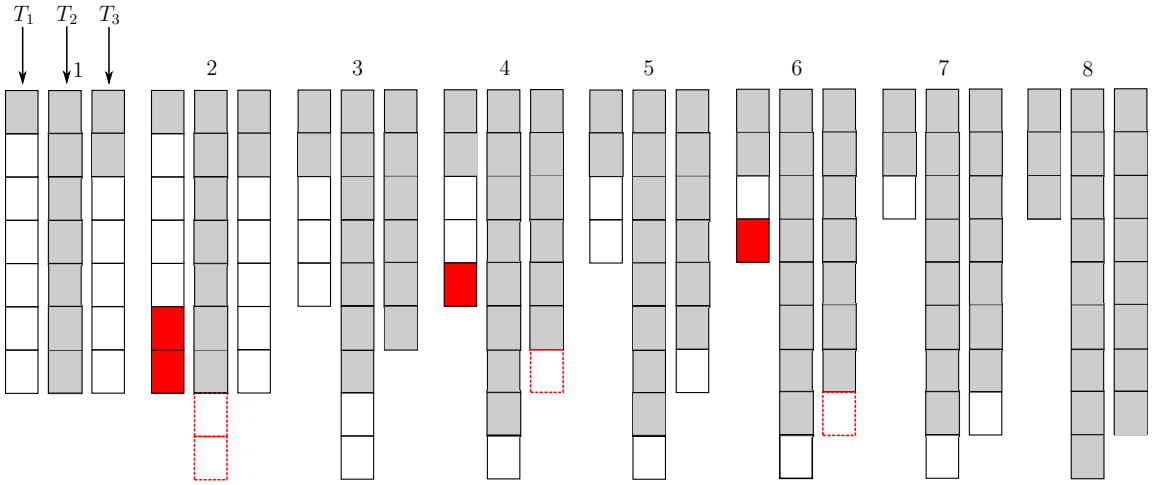
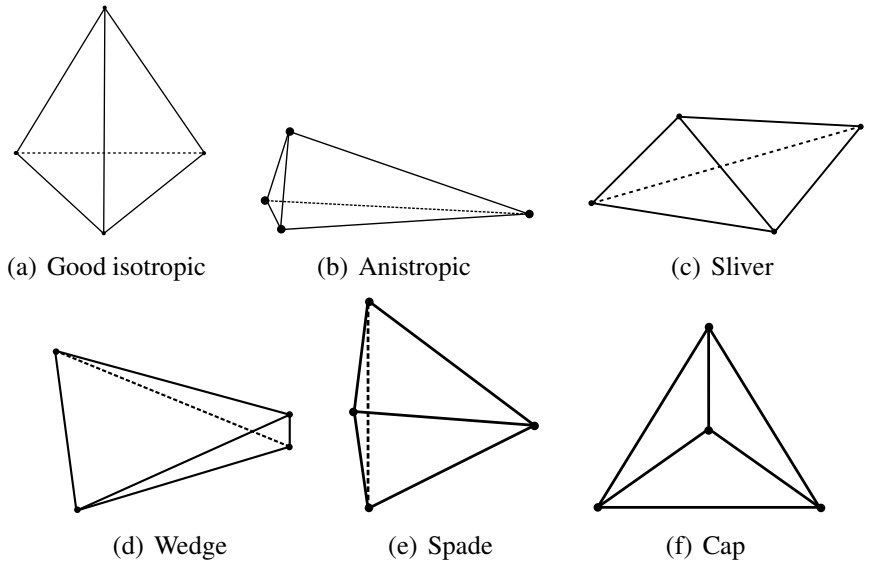


Figure 71: Load balancing with three running threads in eight time steps. Gray indicates the work-units (buckets) currently processed in each time step. White indicates the work-units that have not been processed yet, in each time step. Solid red indicates the work-units to be transferred from a busy thread. Stroke red indicates the work-units to be received by a waiting thread. At step 1, each thread owns seven work-units and T_2 is on a waiting state. At step 2, T_2 is awakened by T_1 after T_1 transfers two work-units to T_2. At step 3, T_3 is on a waiting state. At step 4, one work-unit is transferred from T_1 to T_3. At step 5, all threads are busy. At step 6, one work-unit is transferred again from T_1 to T_3. At step 7, all threads are busy. At step 8, all threads have completed the processing of all work-units.

Unstructured grids contain elements whose shape vary significantly. The problem is more severe in 3-dimensions because tetrahedra can be distorted in more ways compared to triangles in 2-dimensions. The shape of the elements is critical for the accuracy and the convergence of a finite element solution. For example, elements with large dihedral angles tend to increase the discretization error in the solution [8]. On the other hand, elements with small dihedral angles are bad for matrix conditioning but not for interpolation or discretization [74, 173].

Grid generation is followed by a quality improvement step to ensure that the shape of the elements meets the requirements of a specific application. For example, a structural analysis typically requires isotropic elements with shape as close as possible to the regular, i.e., without too small or too large dihedral angles (Figure 72(a)). On the other hand, a computational fluid dynamics analysis for viscous flow requires anisotropic elements stretched in a direction normal to viscous surfaces [14, 131] (Figure 72(b)). This thesis is concerned with isotropic grid generation, thus the grid is improved so that the shape of the elements is as close as possible to regular (Figure 72(a)).

A flat tetrahedron whose vertices are almost coplanar is not suitable for some finite element applications. This tetrahedron is called *sliver* (Figure 72(c)). A sliver may have good planar angles, but it has four small and two large dihedral angles. Figures 72(d)-72(f) depict other types of badly shaped tetrahedra [33]. The wedge has one small dihedral angle, the spade has two small and two large dihedral angles, and the cap has three small and three large dihedral angles. Figure 73 depicts the badly shaped elements within a isotropic tetrahedral grid without quality improvement.



(a) Good isotropic (b) Anistropic (c) Sliver

(d) Wedge (e) Spade (f) Cap

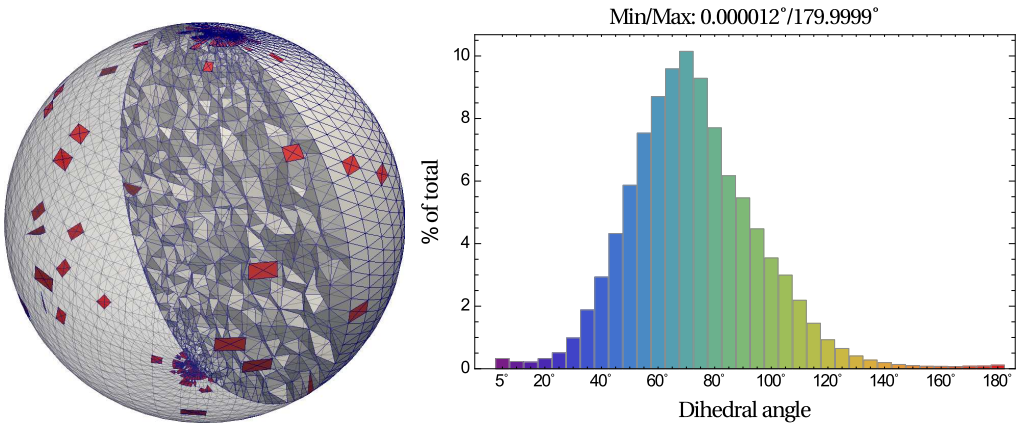
Figure 72: Few possible shapes of a tetrahedron element. (b)-(f) depict tetrahedra with a small or a large dihedral angle.

4.9.1 PROCEDURE

The proposed procedure improves the grid by using:

1. Parallel connectivity optimization with combination of topological transformations.
2. Centroid average or optimal point placement smoothing.
3. Removal of isolated slivers with heuristics.

The grid is improved in $nqual \in [1, 10]$ passes. The default value is $nqual = 3$. Before the first pass, all elements are made active. In each pass, the three operations above are performed. At the beggining of each pass, elements that satisfy a maximum dihedral angle bound $angqual \in [70°, 180°]$ are deactivated. The default bound is $angqual = 120°$.



(a) (b)

Figure 73: Tetrahedral grid of a sphere without quality improvement. The grid contains 352 bad shape elements (shown in red) (a). Each bad shape element has a dihedral angle smaller than $2°$ or larger than $178°$. (b) depicts the element dihedral angle distribution (in 5-deg increments) of the grid and the angle extrema. CDT3D incorporates effective quality improvement techniques that remove the bad shape elements.

Reconnection is attempted only on those elements that do not satisfy the angle bound. The parallel reconnection is presented in subsection 4.8. The smoothing and the removal of isolated slivers is presented in the following paragraphs.

4.9.2 SMOOTHING

Two approaches are implemented to relocate the vertices of the grid: (i) Laplacian-type [69, 132], and (ii) Optimal point placement [1, 132]. Optimal point placement is the default method.

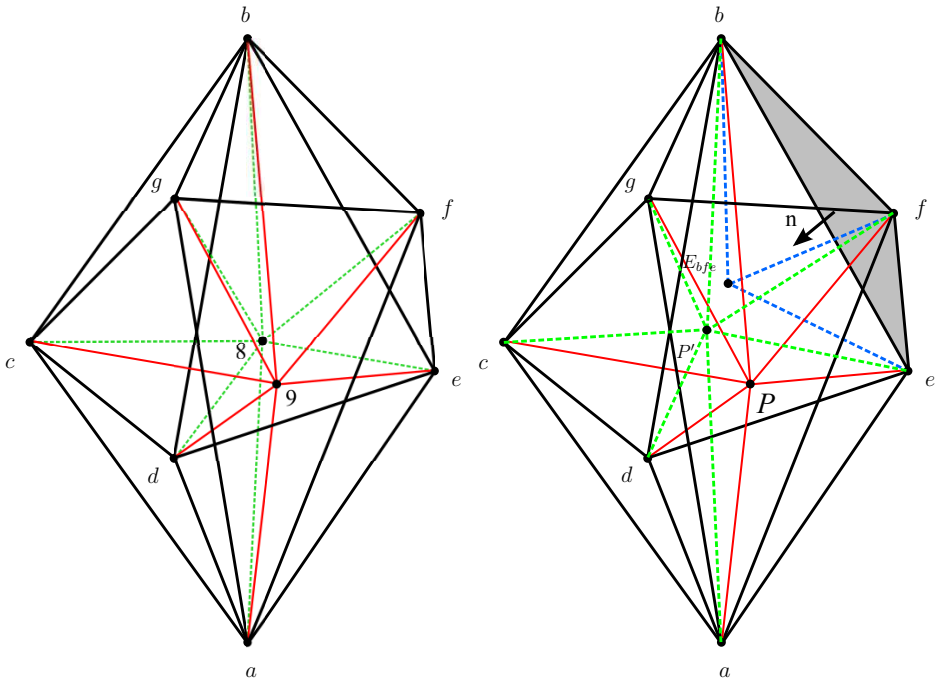
Laplacian-type

The Laplacian-type smoothing repositions each point to an average centroid computed from the tetrahedra surrounding the point (Figure 74(a)):

$$P' = P + c\left[\frac{\sum_{i=1}^{n}(E_i - P)}{n}\right] \tag{9}$$

where P and P' is the point coordinate vector before and after reposition, respectively, c is the smoothing coefficient, n is the number of elements surrounding P, and E_i is the centroid coordinate vector of an element incident on P. A smoothing coefficient of 0.5

is used for all interior points, 0.25 for all points adjacent to boundaries, and zero for all boundary points.



(a) Centroid average (b) Optimal placement

Figure 74: Point smoothing. P and P' are the point coordinate vectors before and after reposition, respectively. P' is the average of the centroids of the tetrahedrons surrounding P (a), or the average of the optimal placements advancing from the boundary faces of a polyhedron formed by the tetrahedrons surrounding P (b). All boundary faces of the polyhedron are visible from P because the point is interior and a valid grid is maintained throughout the grid generation. For simplicity, (b) depicts one optimal position E_{bfe} advancing from face bfe.

Optimal point placement

This approach repositions each point to the average of n optimal positions, where n is the number of boundary faces of a polyhedron formed by the tetrahedra surrounding the point (Figure 74(b)). P' is calculated from (9), where E_i is the coordinate vector of an optimal position advancing from boundary face i. The coordinate vector E_{bfe} in Figure 74(b) is:

$$E_{bfe} = C_{bfe} + \sqrt{\frac{2}{3}} \cdot \frac{\vec{n}}{|\vec{n}|} \cdot \frac{1}{3} \left(|\vec{bf}| + |\vec{fe}| + |\vec{eb}| \right) \tag{10}$$

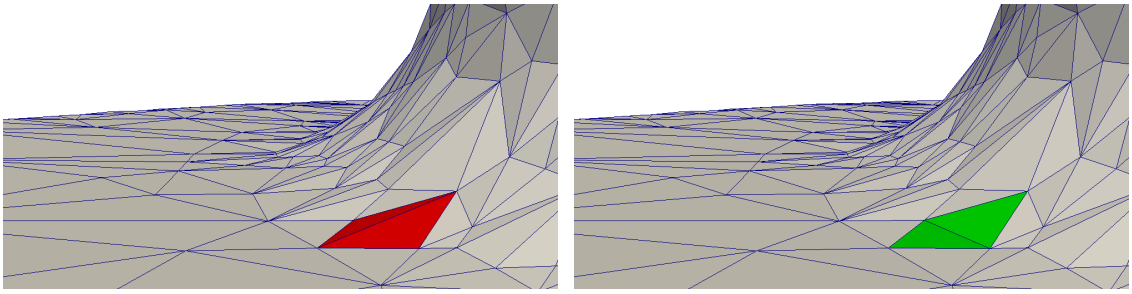
where C_{bfe} and $\vec{n}$ is the centroid and the normal of face bfe, respectively.

Two iterations are performed for the selected smoothing method. If smoothing creates inverted elements, the coefficient c is scaled with 0.5 and smoothing is attempted again.

4.9.3 SLIVER REMOVAL WITH HEURISTICS

Direct sliver removal

Isolated sliver elements with two faces on the boundary surface are directly removed (Figure 75). The sliver should have at least one dihedral angle greater than $angdbs \in [120°, 180°]$. The default value is $angdbs = 160°$. Direct removal does not preserve the boundary connectivity, thus it is not suitable for applications which require fixed boundaries. This option is activated with the boundary sliver deletion flag $mdbs$.



(a) Before removal (b) After removal

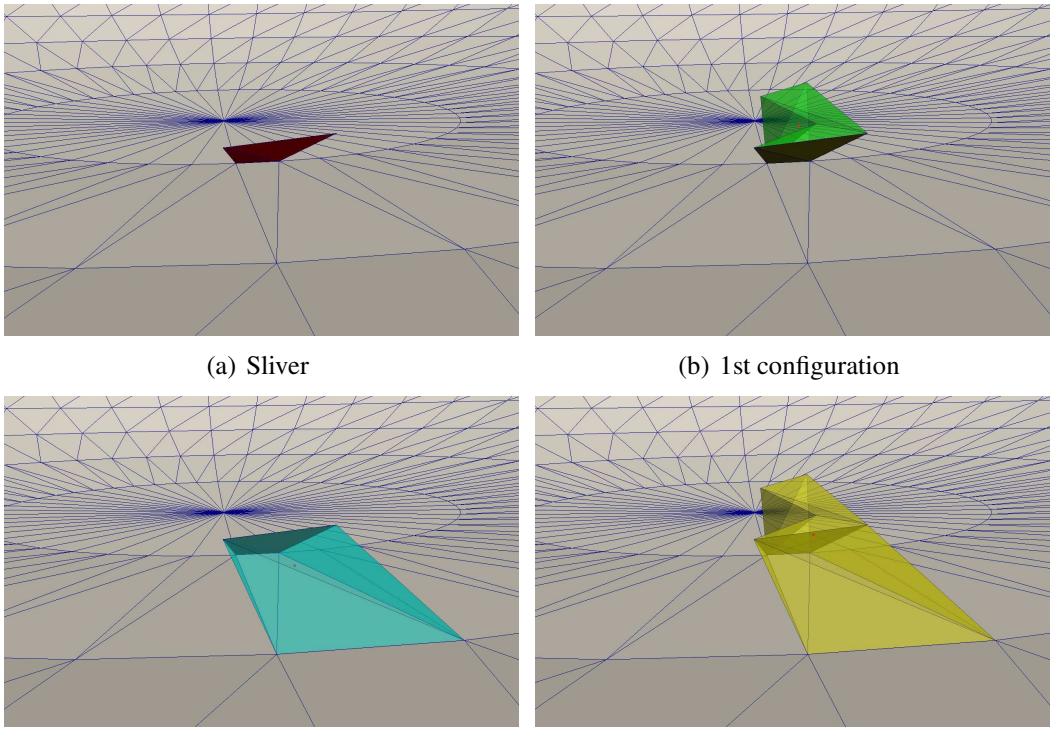
Figure 75: Direct removal of an isolated sliver element with two boundary faces.

Sliver removal by point insertion and local reconnection

This approach inserts a new point inside a polyhedron near the sliver and locally reconnects the polyhedron to form high quality elements (Figure 76). An element is considered as a field sliver if it has at least one dihedral angle greater than $angdfs \in [120°, 180°]$. The default value is $angdfs = 165°$. This option is activated with the quality field sliver deletion flag $mdfs$.

If all the faces of the sliver are interior (Figure 76(a)), the method computes three candidate configurations (Figures 76(b)-76(d)). Otherwise the method computes one candidate configuration. The configuration that minimizes the maximum dihedral angle is selected. The polyhedron in the 1st and 2nd configuration contains all the tetrahedra surrounding an edge of the sliver with a dihedral angle larger than $angdfs$ (Figures 76(b),76(c)). The polyhedron in the 3rd configuration is the union of the polyhedrons of the other two configurations (Figure 76(d)).

A new point is inserted at the centroid of the polyhedron. The point is connected with the points on the boundary of the polyhedron. If the reconnection is not geometrically valid, then the point is inserted at the centroid of the sliver. If the reconnection is still not geometrically valid, then the operation is postponed until the next quality improvement pass. If reconnection is geometrically valid but creates new slivers (tetrahedra with dihedral angle greater than $angdfs$), the point coordinates are smoothed using the methods described in subsection 4.9.2. If smoothing does not satisfy the quality bounds, the insertion is postponed until the next improvement pass. In general, the removal of a sliver element cannot be guaranteed. In practice, given a surface grid of a reasonable good shape triangles, the combination of smoothing, local reconnection, and point insertion eliminates all slivers within three improvement passes.



(a) Sliver (b) 1st configuration

(c) 2nd configuration (d) 3rd configuration

Figure 76: Configurations to remove isolated slivers with point insertion and local reconnection. (a) depicts a sliver inside the domain. The 1st and the 2nd configuration contain the tetrahedra (including the sliver) surrounding an edge of the sliver with a dihedral angle larger than $angdfs$. The 3rd configuration is the union of the 1st and the 2nd configuration. The configuration which maximizes the quality after point insertion and reconnection is selected. The new point is attempted to insert at the centroid of the polyhedron and reconnect with the boundary points of the polyhedron. Alternatively, the point can be inserted at the centroid of the sliver or it can be smoothened to satisfy quality bounds.

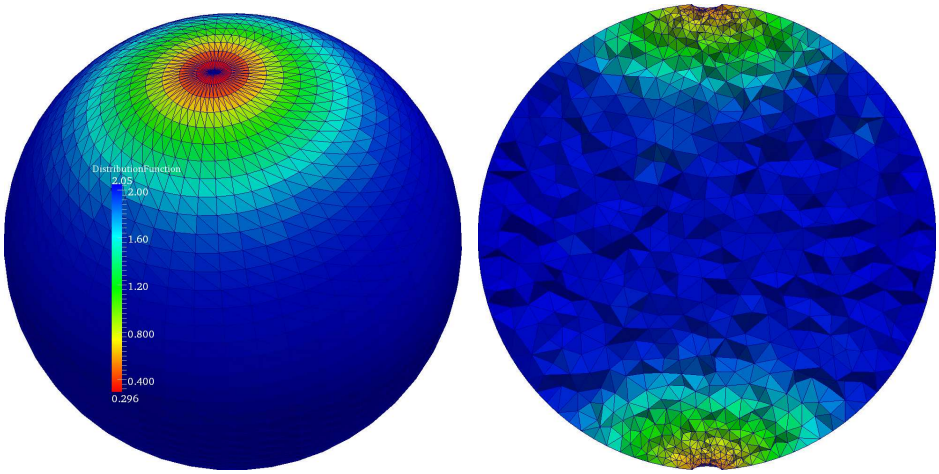
4.10 EVALUATION RESULTS

This thesis conducts an extensive evaluation on quality, execution time and scalability of the proposed algorithm. The evaluation is performed on geometries pertinent to aerospace applications.

4.10.1 FIELD POINT SPACING

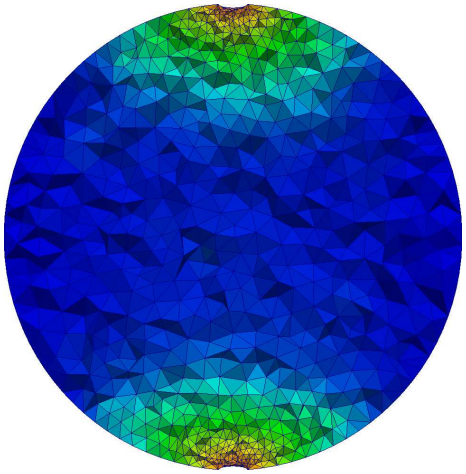
CDT3D adopts a distribution function weighting factor $cdfm$ to control the field point spacing [132]. Increasing $cdfm$ above zero will generally reduce the growth of spacing from small to larger tetrahedra and increase the total number of grid points generated. Two experiments are conducted with various weighting factors to demonstrate the element growth rate. The first experiment is conducted on a sphere (Figure 77) and the second on two blades with merging wakes and symmetry plane inside a field (Figure 78). Note that the wake region in Figure 78(a) is used only to control the field point spacing. The input surface grids obtained from the CAVS Sims Center at MSU. Figures 79 and 80 depict the quality histograms of the volume grids. All tetrahedral grids are of high quality.

The distribution function is initially computed from the average of the boundary edge lengths surrounding a boundary point. CDT3D employs a maximum distribution function $dfmax$ to limit the point spacing in the field [132]. If $dfmax < 0$ then the distribution function is not limited. If $dfmax = 0$ then the distribution function is limited to the maximum value determined from the boundary surface grid. If $dfmax > 0$ then the distribution function is limited to $dfmax$. Figure 81 depicts volume grids generated for varied $dfmax$. The surface grid obtained from the CAVS Sims Center at MSU. A limited distribution function can produce a uniform fild point spacing (Figures 81(c) and 81(d)).

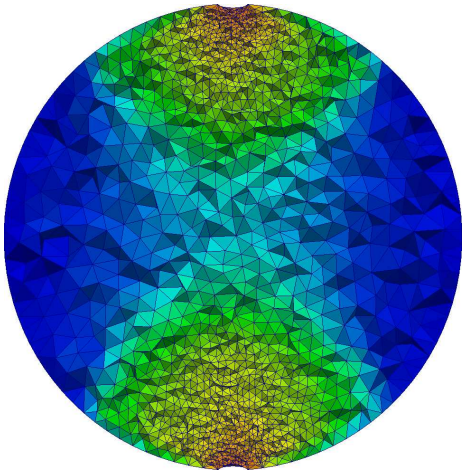


(a) Input surface

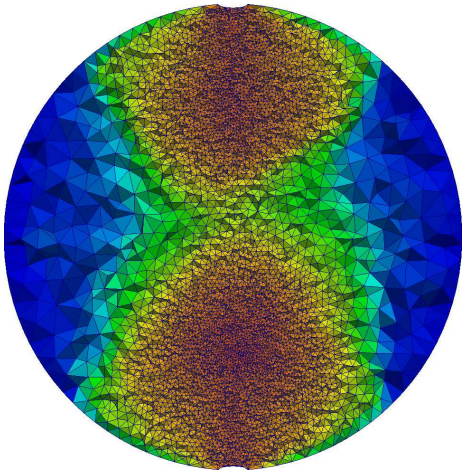
(b) $cdfm$: 0; #tets : 57K



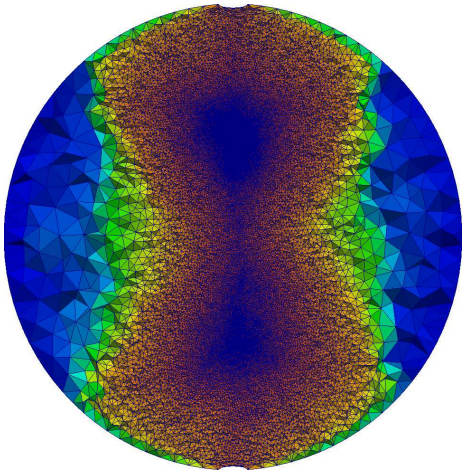
(c) $cdfm$: 0.2; #tets : 58K



(d) $cdfm$: 0.6; #tets : 129K

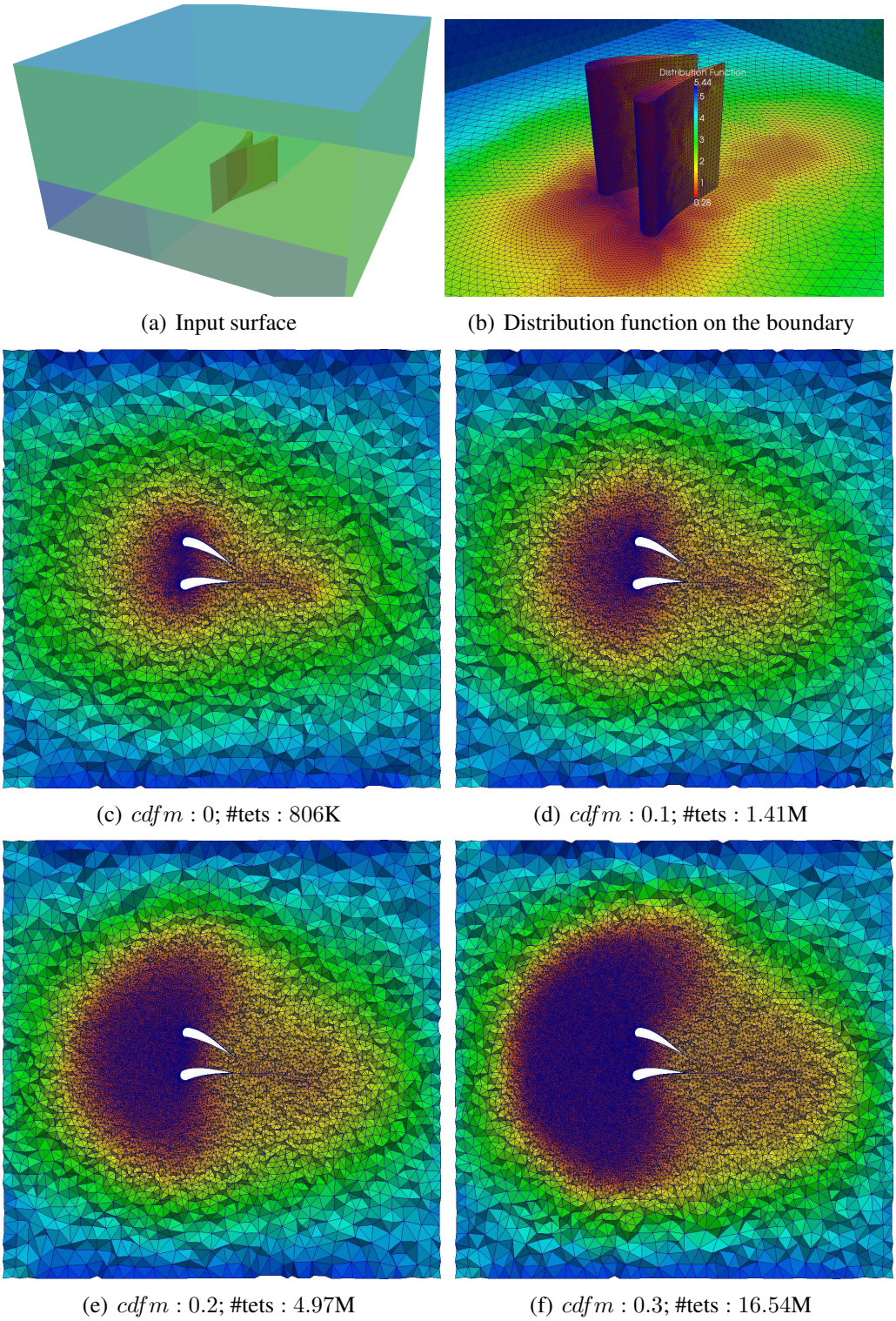


(e) $cdfm$: 0.8; #tets : 854K



(f) $cdfm$: 0.9; #tets : 7.68M

Figure 77: Cuts of tetrahedral grids of a sphere generated for varied $cdfm \in [0, 1]$. Blue corresponds to larger values of the distribution function. Red corresponds to smaller values of the distribution function.



(a) Input surface

(b) Distribution function on the boundary

(c) $cdfm$: 0; #tets : 806K

(d) $cdfm$: 0.1; #tets : 1.41M

(e) $cdfm$: 0.2; #tets : 4.97M

(f) $cdfm$: 0.3; #tets : 16.54M

Figure 78: Cuts of tetrahedral grids of two blades with merging wakes and a symmetry plane enclosed in an outer boundary generated for varied $cdfm \in [0, 1]$ (shown in (c)-(f)). The input surface is depicted in (a)-(b). The wake region is modeled as an embedded/transparent delete surface.

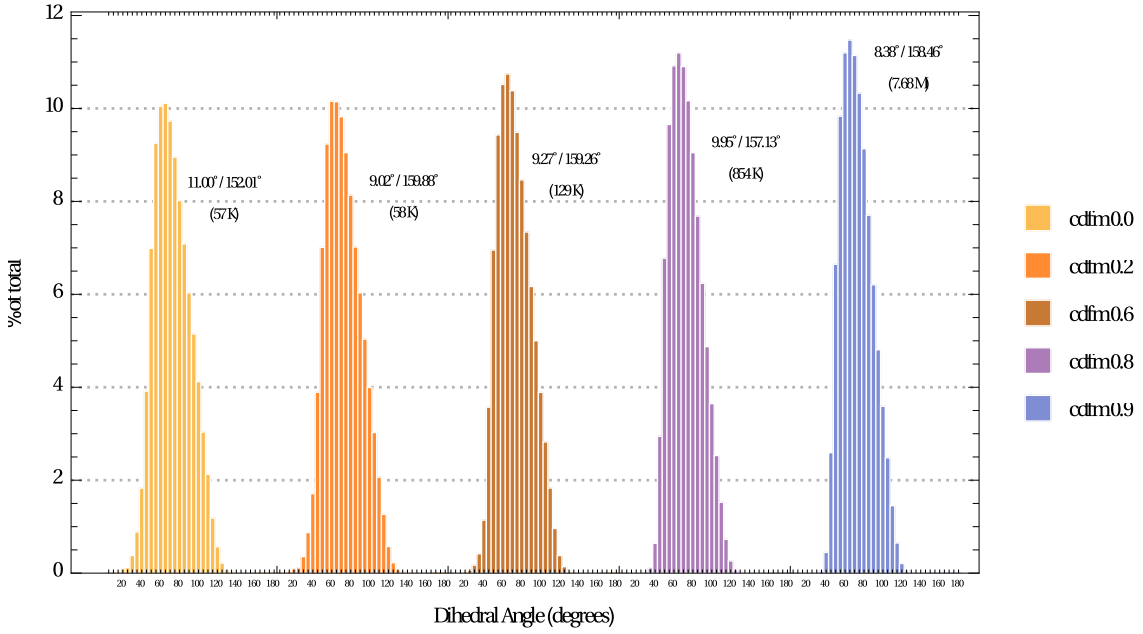


Figure 79: Element angle distribution (in 5-deg increments) of grids of sphere, for varied $cdfm$. The dihedral angle extrema and the element count are reported for each grid.

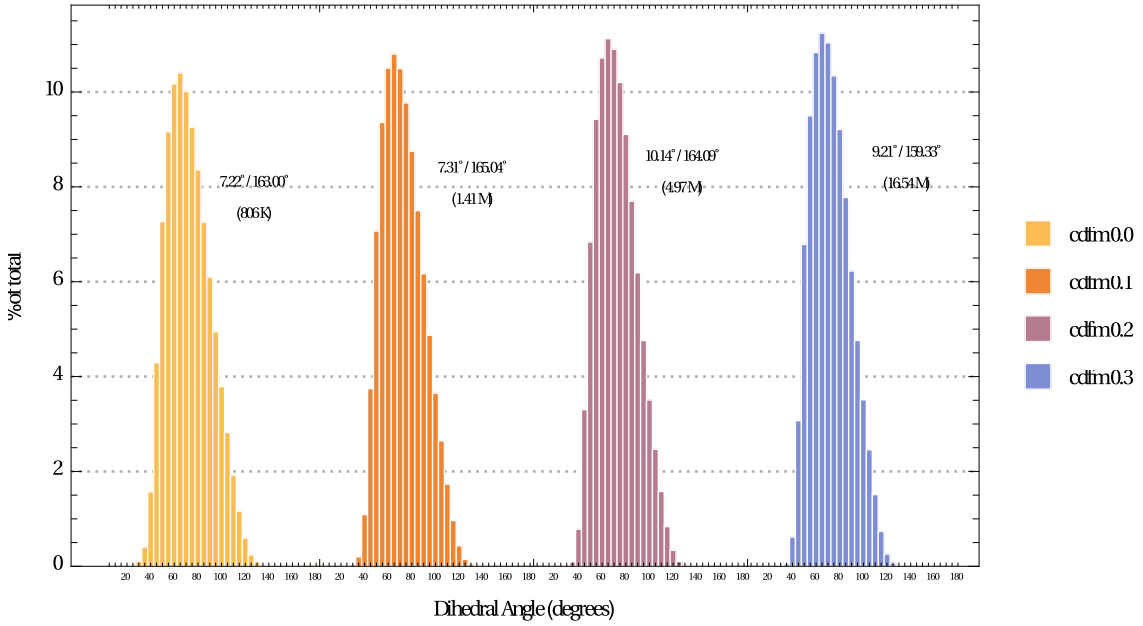
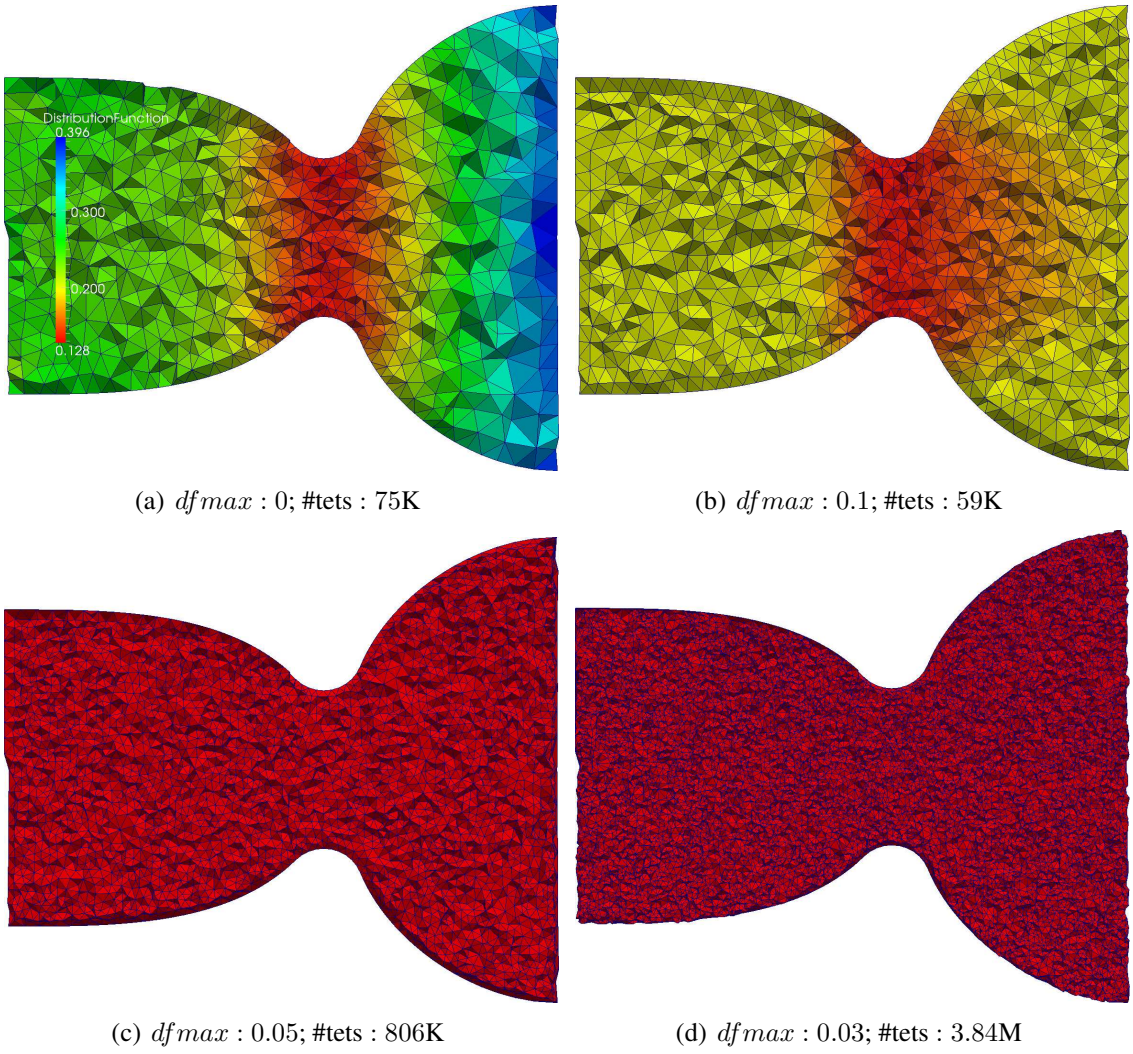


Figure 80: Element angle distribution (in 5-deg increments) of grids of two blades with merging wakes and a symmetry plane enclosed in an outer boundary, for varied $cdfm$. The dihedral angle extrema and the element count are reported for each grid.



(a) $dfmax$: 0; #tets : 75K

(b) $dfmax$: 0.1; #tets : 59K

(c) $dfmax$: 0.05; #tets : 806K

(d) $dfmax$: 0.03; #tets : 3.84M

Figure 81: Cuts of tetrahedral grids of a nozzle geometry (Figure 63(f)) generated for varied $dfmax \in [-1, 1e+19]$.

4.10.2 COMPARISON WITH AFLR

CDT3D is compared with state-of-the-art unstructured grid technology AFLR v16.9 [132, 131]. AFLR is directly incorporated in several systems, including: DoD CREATE-MG Capstone, Lockheed Martin/DoD ACAD, Boeing MADCAP, MSU SolidMesh, and Altair HyperMesh. CDT3D and AFLR have a handful of options for quality grid generation and optimization. Only a few basic options of these codes are tested, hence these comparisons are far from comprehensive. The comparison is performed on three realistic aerospace configurations:

1. An aircraft nacelle with engine inside a section of wind tunnel (Figure 82).

2. A rocket with engine, nozzle, and transparent internal data surfaces inside a flow field (Figure 83).

3. A launch vehicle with solid boosters inside a flow field (Lv2b) (Figure 84).

The surface grids of the geometries obtained from CAVS Sims Center at MSU in .surf format. Tetrahedral grids are attempted to generate using TetGen v1.5.0 [179] but TetGen failed to preserve the input surfaces into the volume grids. Therefore, TetGen's results are omitted.

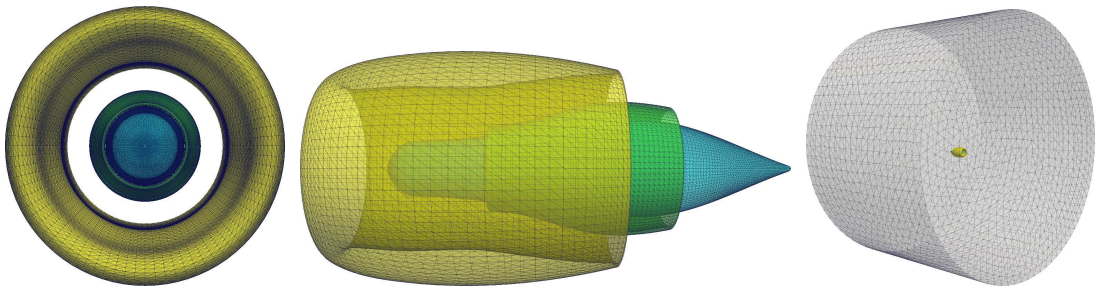


Figure 82: Surface grid of an aircraft nacelle with engine inside a section of wind tunnel. #points: 27184; #triangles: 54360.

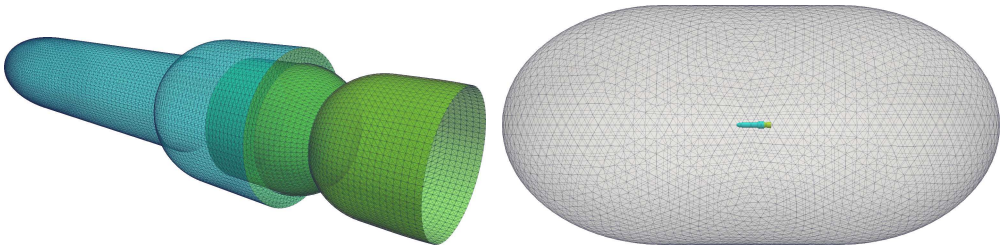


Figure 83: Surface grid of a rocket with engine, nozzle and transparent internal data surfaces inside flow field. #points: 20228; #triangles: 40448.

Table 17 lists the parameters for the evaluation. $cdfm$ controls the growth of element size from small to larger elements (it is adjusted so that the two codes will generate approximately the same number of elements); $cdfn$ is the nearby point factor; $mrecm$ is the local-reconnection flag; $mrecm : 2$ uses a combined Delaunay and Min-Max type criterion for local reconnection; $nqual$ is the number of quality improvement passes; $csmth$ is the smoothing coefficient; $msmth : 1$ uses an optimal point placement for smoothing; $nsmth$ is the number of smoothing iterations; $angdfs$ is the quality field sliver dihedral

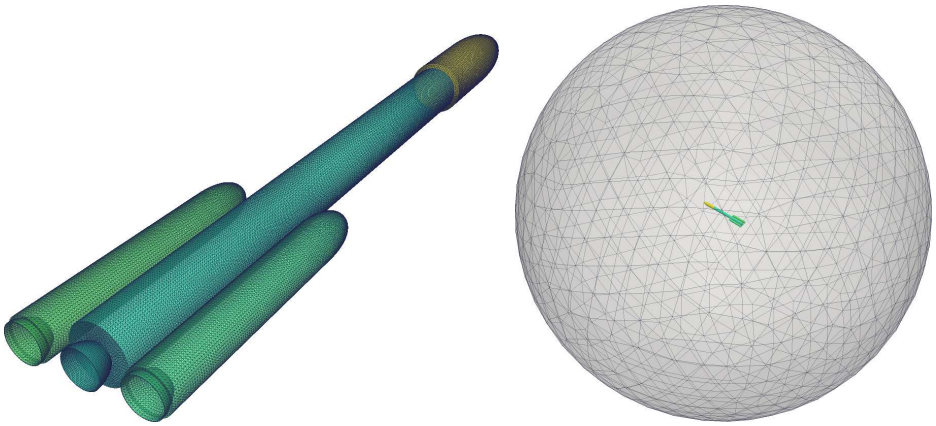


Figure 84: Surface grid of a launch vehicle with solid boosters inside flow field (Lv2b). #points: 42020; #triangles: 84024.

angle threshold; $angqual$ is the final quality dihedral element angle threshold; $mdbs$: 0 prevents the reconnection of boundary surface triangles during sliver deletion; $nthreads$ is the number of the threads for parallel reconnection (only CDT3D); $nbuckets$ is the granularity of the decomposition (only CDT3D); $frbtransf$ is the fraction of work which is transferred among threads to balance the work load (only CDT3D); $cbtransf$ is a coefficient to reduce the amount of work which is transferred between two threads (only CDT3D); $mrecrbf$: 0 turns off the reconnection of boundary surface faces to preserve the boundary connectivity (only AFLR).

Table 17: Parameters for unstructured grid generation. Additional parameters only for CDT3D: $nthreads$: 12 (parallel); $nthreads$: 1 (sequential); $nbuckets$: 240; $frbtransf$: 0.3; $cbtransf$: 1.0. Additional parameters only for AFLR: $mrecrbf$: 0. Appendix B.3 lists the CDT3D parameters.

Geometry	Software	Refinement				Improvement					
		$cdfm$	$cdfn$	$mrecm$	$nqual$	$csmth$	$msmth$	$nsmth$	$angdfs$	$angqual$	$mdbs$
Nacelle	CDT3D	0.291	0.7	2	3	0.5	1	2	165°	120°	0
	AFLR	0.50									
Rocket	CDT3D	0.20									
	AFLR	0.60									
Lv2b	CDT3D	0.234									
	AFLR	0.30									

Table 18 presents the results. This study uses the dihedral angle as a metric to evaluate the element quality. The dihedral angle is determined from the normalized inner product of the two face area vectors for the two element faces with a common element edge. Both

Table 18: Evaluation results on unstructured grid generation. CDT3D is compared with state-of-the-art technology AFLR v16.9.19 [132]. CDT3D's runs are performed with 1 and 12 hardware cores. AFLR is a sequential code. Table 17 lists the parameters of the evaluation. The sliver elements have a dihedral angle smaller than 2° or larger than 178°. Initial grid includes Delaunay tetrahedralization and Boundary Recovery. The I/O time is not included. The experiments performed on a DELL workstation with Linux Ubuntu 12.10, 12 cores Intel(R) Xeon(R) CPU X5690@3.47 GHz, and 96 GB RAM.

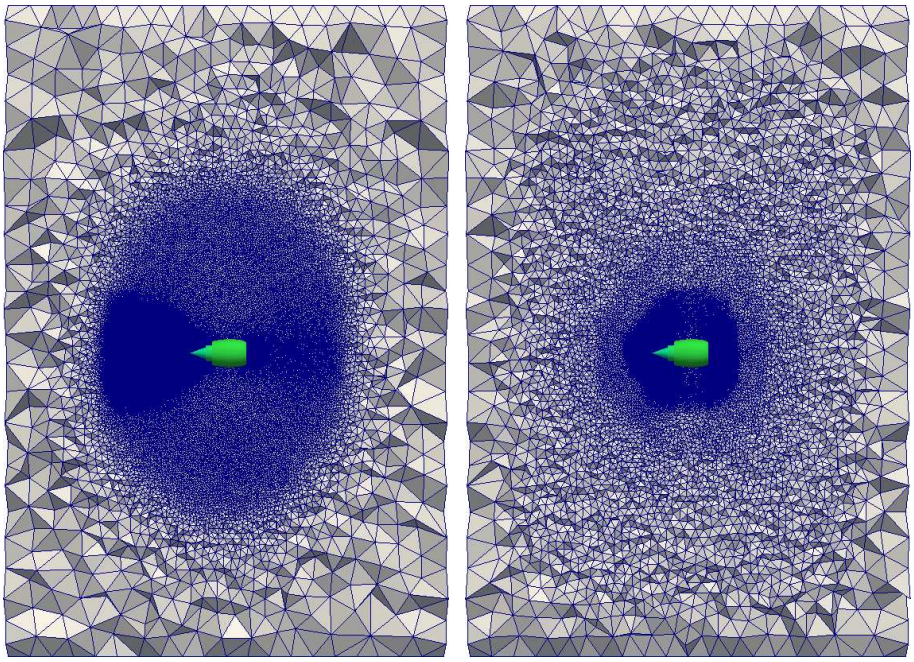
Case	Software	#Cores	%Slivers (w/o improv.) ($\times 10^{-3}$)	#Tets (w/ improv.) (M)	Min/Max Angle (w/ improv.) (deg)	Time Initial Grid (sec)	Time Refinement (min)	Time Improvement (min)	Time Total (min)
Nacelle	CDT3D	1	3.74	43.65	$13.57^\circ/153.44^\circ$	1.36	20.01	14.30	34.33
		12	3.70	42.85	$12.06^\circ/159.52^\circ$	1.36	5.02	18.59	23.64
	AFLR	1	2.97	43.16	$7.00^\circ/164.86^\circ$	5.63	22.59	6.40	29.09
Rocket	CDT3D	1	2.96	118.41	$9.39^\circ/159.30^\circ$	1.58	52.85	64.56	117.44
		12	2.95	119.06	$9.21^\circ/158.33^\circ$	1.58	14.51	68.23	82.76
	AFLR	1	3.05	123.13	$5.58^\circ/164.75^\circ$	6.76	131.89	25.41	157.42
Lv2b	CDT3D	1	5.09	98.21	$6.60^\circ/159.68^\circ$	5.45	41.57	94.63	136.29
		12	4.69	113.99	$8.24^\circ/158.59^\circ$	5.45	12.92	62.36	75.37
	AFLR	1	3.49	104.10	$6.84^\circ/164.88^\circ$	16.97	98.24	18.51	117.03

methods generate grids of good quality elements. CDT3D exhibits a higher quality compared to AFLR in both the sequential and the parallel runs. CDT3D eliminates all elements whose dihedral angles are smaller than 6.60° or larger than 159.68°. The corresponding values for AFLR are 5.58° and 164.86°, respectively (Table 18).

Figures 85, 86, 87, and 88 depict cuts of the grids. Figures 89 and 90 depict the element angle distribution. For a more comprehensive comparison, Figure 89 compares the element angle distribution before and after a quality improvement step. At the completion of the refinement a small percentage of sliver elements may survive $(< 0.003\%)$. CDT3D incorporates effective quality improvement techniques to eliminate the sliver elements. Figure 91 depicts the low quality elements after the completion of the refinement of the aircraft nacelle.

CDT3D completes the end-to-end grid generation (including initial grid construction, refinement, and improvement) up to 1.90 times faster than AFLR, when 12 hardware cores are utilized (Table 18). Most notably, CDT3D refines the grids up to 2.5 and 10 times faster compared to AFLR, when 1 and 12 hardware cores are utilized, respectively. On the other hand, AFLR exhibits very good performance at the improvement step but the grids are of a lower quality compared to CDT3D. Both methods complete the construction of the initial grid (i.e., Delaunay tetrahedralization and boundary recovery) at a negligible cost (less than 1% of the total generation time).

Figures 92-93 depict the number of topological transformations for the sequential optimization of Lv2b grid using a combined criterion (Table 18). In each refinement pass, the connectivity is optimized in two phases. In the first phase reconnection is performed with a Delaunay in-sphere criterion (Figure 92). In the second phase reconnection is performed with a Min-Max type criterion (Figure 93). The first phase requires higher iteration counts compared to the second phase. This is because the Delaunay criterion improves dramatically the distribution of dihedral angles after direct point insertion, at the expense of degrading the extremal angles. The Min-Max criterion follows to improve the extremal angles.



(a) CDT3D: 42.85 M tetrahedra (b) AFLR: 39.64 M tetrahedra

Figure 85: Tetrahedral field cuts of the aircraft nacelle.

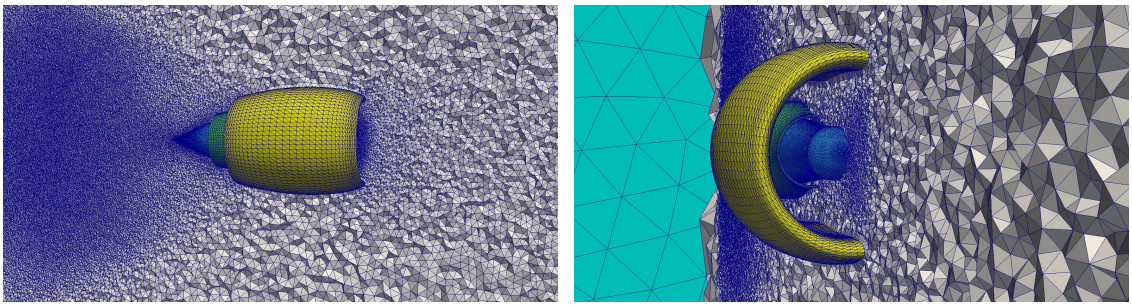
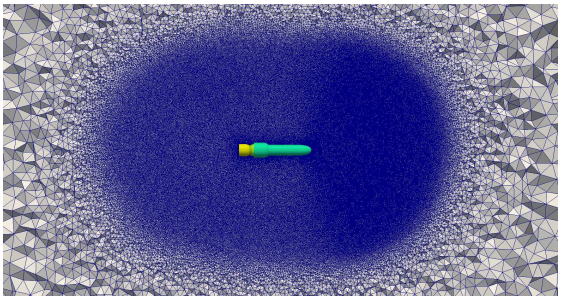
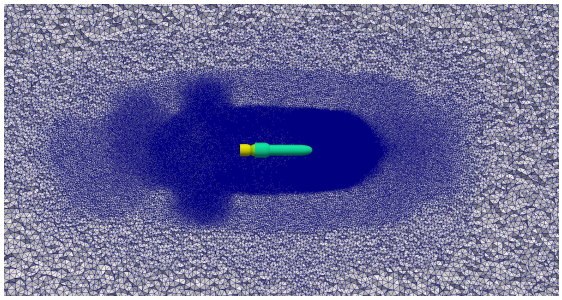


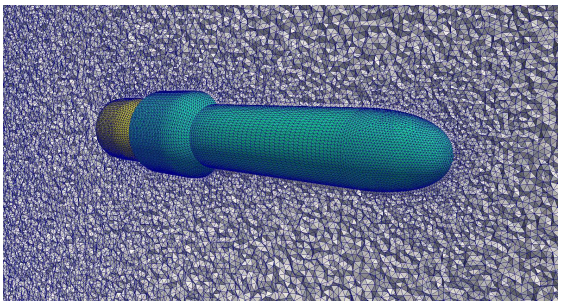
Figure 86: Detrail views of tetrahedral field cuts of aircraft nacelle generated with CDT3D.



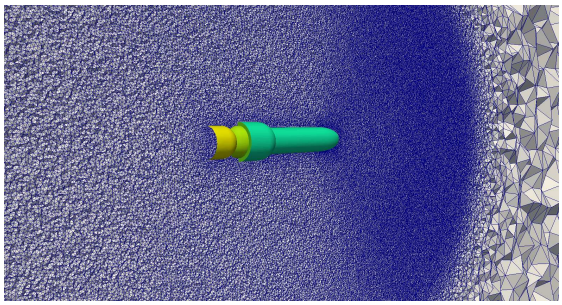
(a) CDT3D: 119.06 M tetrahedra



(b) AFLR: 127.68 M tetrahedra

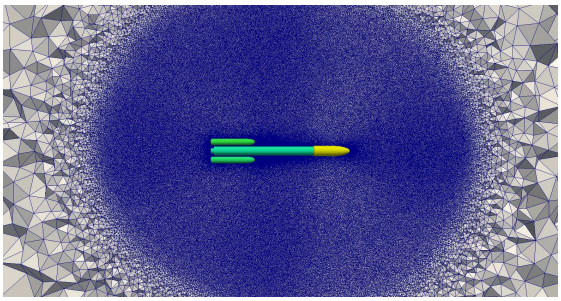


(c) Detail view of (a)

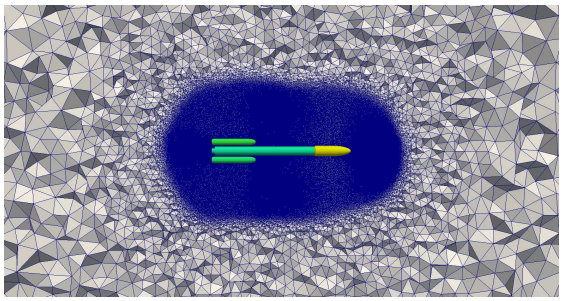


(d) Detail view of (a)

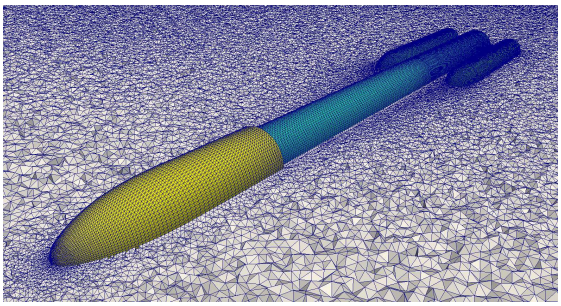
Figure 87: Tetrahedral field cuts of the rocket.



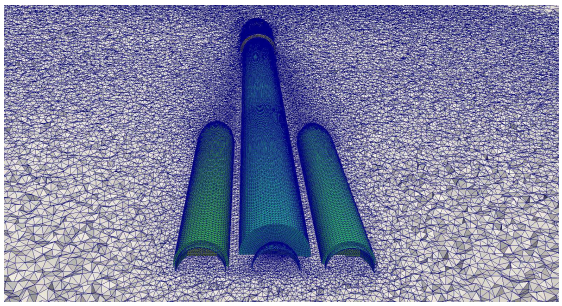
(a) CDT3D: 110.01 M tetrahedra



(b) AFLR: 100.92 M tetrahedra

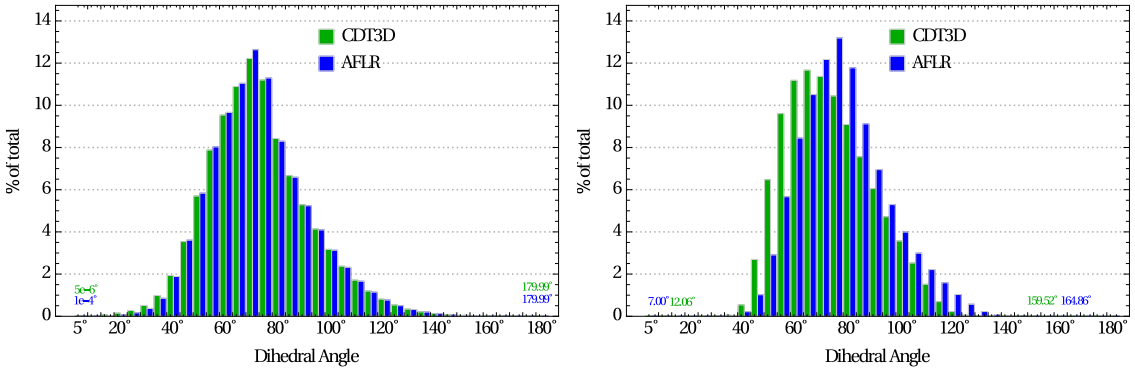


(c) Detail view of (a)



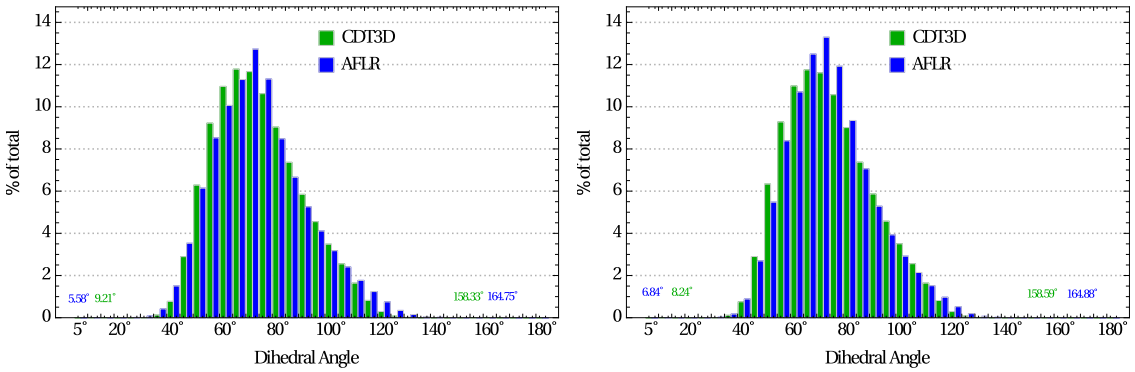
(d) Detail view of (a)

Figure 88: Tetrahedral field cuts of the Lv2b.



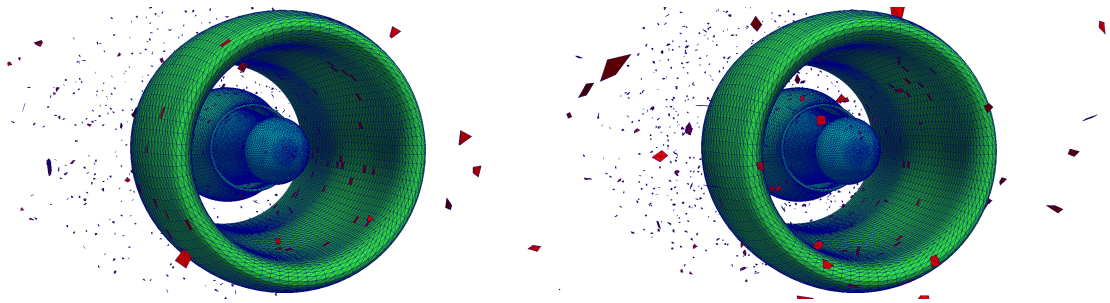
(a) w/o improvement (b) w/ improvement

Figure 89: Element angle distribution (in 5-deg increments) of aircraft nacelle grids. The dihedral angle extrema are reported for each method.



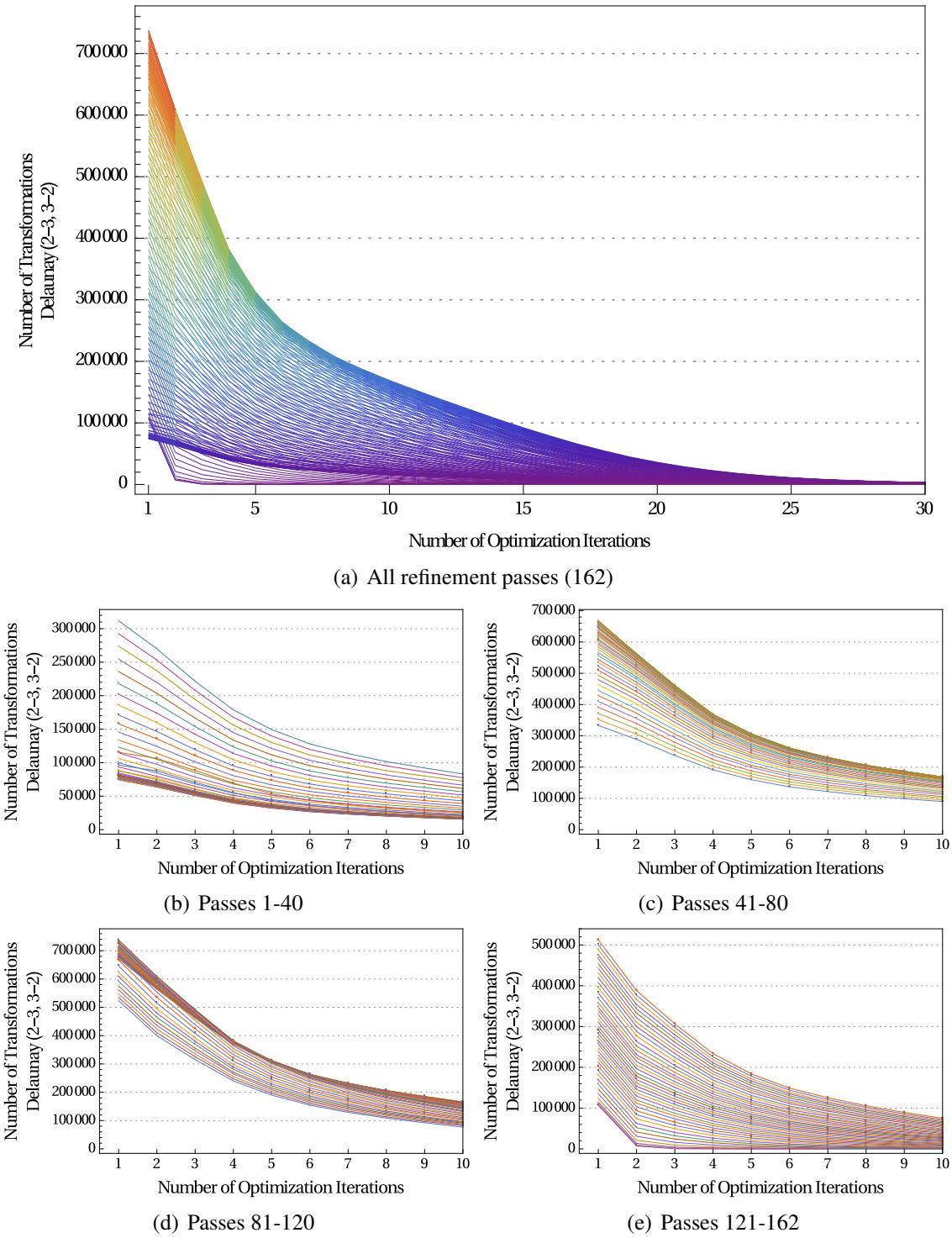
(a) Rocket (w/ improvement) (b) Lv2b (w/ improvement)

Figure 90: Element angle distribution (in 5-deg increments) after improvement of Rocket and Lv2b grids. The dihedral angle extrema are reported for each method.



(a) CDT3D: 1590 slivers (0.0037%) out of total 42.91 M tetrahedra (b) AFLR: 1287 slivers (0.0029%) out of total 43.23 M tetrahedra

Figure 91: Slivers after the completion of refinement of the aircraft nacelle. Red represents elements whose minimum dihedral angle is smaller than $2°$ or larger than $178°$.



(a) All refinement passes (162)

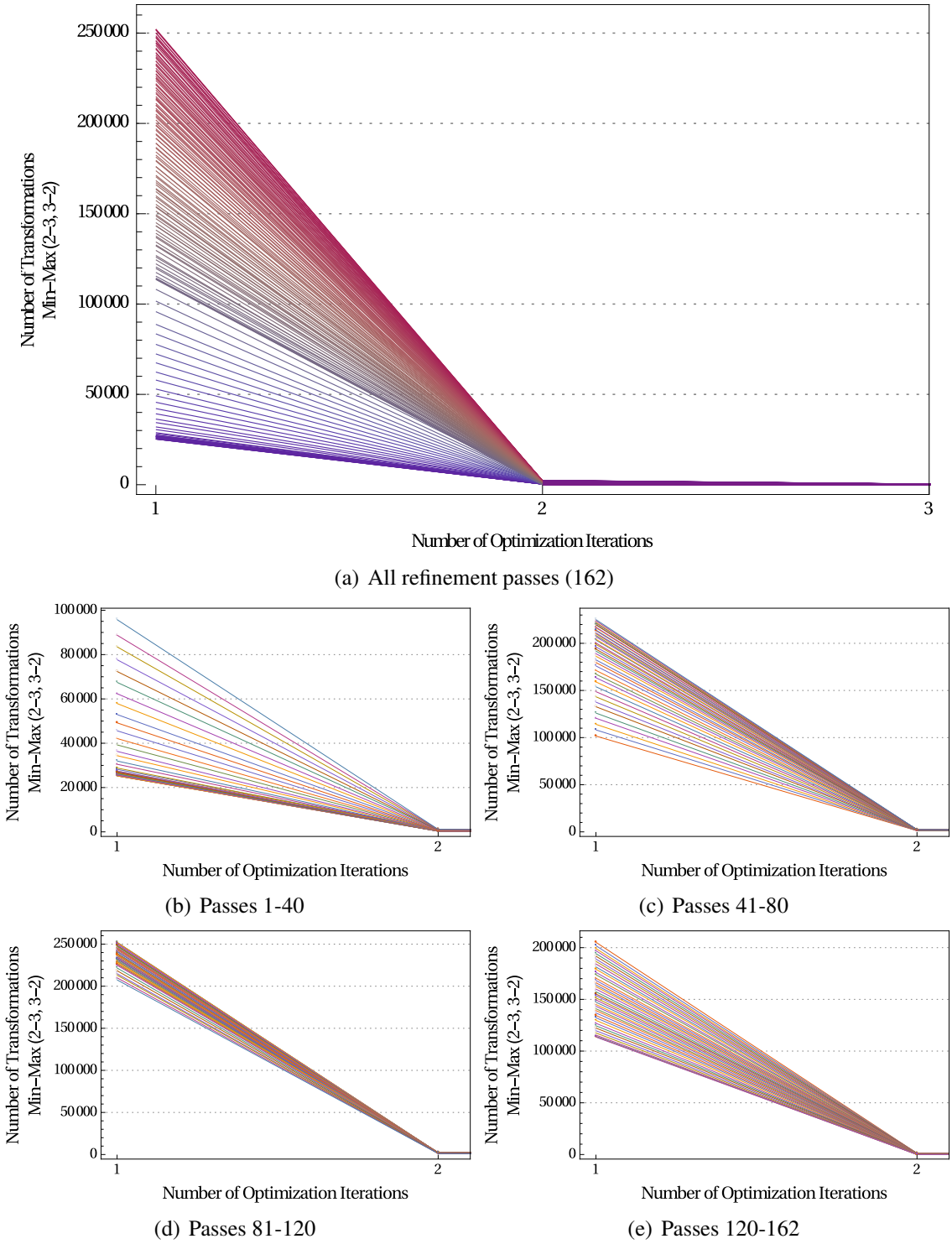
(b) Passes 1-40

(c) Passes 41-80

(d) Passes 81-120

(e) Passes 121-162

Figure 92: Number of topological transformations for connectivity optimization of Lv2b grid (Table 18) using a Delaunay criterion. Each line illustrates a refinement pass. The refinement completes in 162 passes.



(a) All refinement passes (162)

(b) Passes 1-40

(c) Passes 41-80

(d) Passes 81-120

(e) Passes 120-162

Figure 93: Number of topological transformations for connectivity optimization of Lv2b grid (Table 18) using a Min-Max type criterion that follows a Delaunay criterion. Each line illustrates a refinement pass. The refinement completes in 162 passes.

4.10.3 RESULTS FOR VARIED QUALITY CRITERIA AND TRANSFORMATIONS

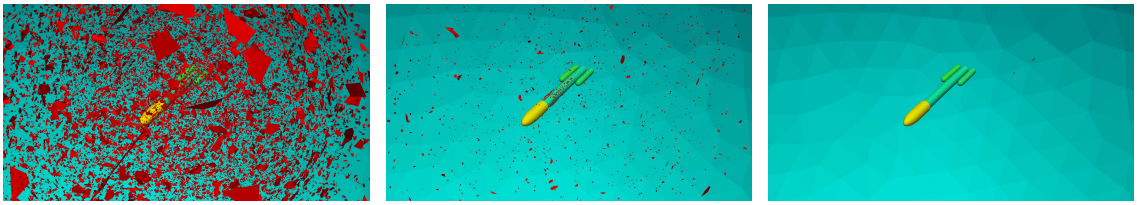
CDT3D optimizes the connectivity by using either a single quality criterion (i.e., Delaunay) or a combined quality criterion (i.e., Delaunay + Min-Max). In the former case, the grid is repetitively reconnected in one pass using combinations of 2-3, and 3-2 flips. In the latter case, the grid is repetitively reconnected in two passes using either combinations of 2-3 and 3-2 flips, or combinations of 2-3, 3-2, and 4-4 flips. The improvement step adopts a combined criterion and combinations of 2-3, 3-2, and 4-4 flips to obtain a maximum quality. Table 19 reports performance results for varied quality criteria and topological transformations.

Table 19: Performance results for the grid generation of the Lv2b geometry, for varied quality criteria and topological transformations. Delaunay: In-sphere criterion; Min-Max: maximization of the minimum Laplacian edge weight. 2-3, 3-2, and 4-4 are flip types. Parameters: $cdfn$: 0.7; $nqual$: 3; $nsmth$: 2; $nthreads$: 12; $nbuckets$: 240; $frbtransf$: 0.3; $cbtransf$: 1.0. Parameters only for Delaunay (2-3, 3-2): $cdfm$: 0.23; $mrecm$: 1. Parameters only for Delaunay (2-3, 3-2, 4-4): $cdfm$: 0.23; $mrecm$: 2. Parameters only for Delaunay + Min-Max (2-3, 3-2, 4-4): $cdfm$: 0.8; $mrecm$: 2; $mrec4$: 1. The experiments performed on a DELL workstation with Linux Ubuntu 12.10, 12 hardware cores Intel(R) Xeon(R) CPU X5690@3.47 GHz, and 96 GB RAM

	Refinement			Improvement				Total
Criteria	#Tets (M)	#Slivers ($< 2°$ or $> 178°$)	Time (min)	#Tets (M)	#Slivers ($< 2°$ or $> 178°$)	Min/Max Dihedral Angle	Time (min)	Time (min)
Delaunay (2-3, 3-2)	84.18	82138	16.15	79.47	106	2.2e-4°/179.99°	149.70	165.85
Delaunay+Min-Max (2-3, 3-2)	86.24	4236	9.62	86.12	0	4.74°/163.17°	88.09	97.71
Delaunay+Min-Max (2-3, 3-2, 4-4)	78.97	62	20.19	78.95	0	12.78°/161.90°	84.39	104.58

The combined criterion produces a grid with quality superior to that of a grid generated using a single criterion. Specifically, a combined criterion with combinations of 2-3, 3-2, and 4-4 flips produces 1324 times fewer slivers at the end of refinement (without improvement) compared to a single criterion with combinations of 2-3 and 3-2 flips. The additional timing cost to achieve this is about 25% (Table 19).

Figure 94 depicts the generated slivers after the completion of the refinement. The improver takes longer to optimize the grid when a single criterion for refinement is employed, and eventually 106 slivers survive (Table 19). The end-to-end generation time (including improvement) reduces by about 37% and 41% when a combined criterion with 2-3, 3-2, and 4-4 flips or 2-3 and 3-2 flips is used, respectively, compared to a single criterion. The

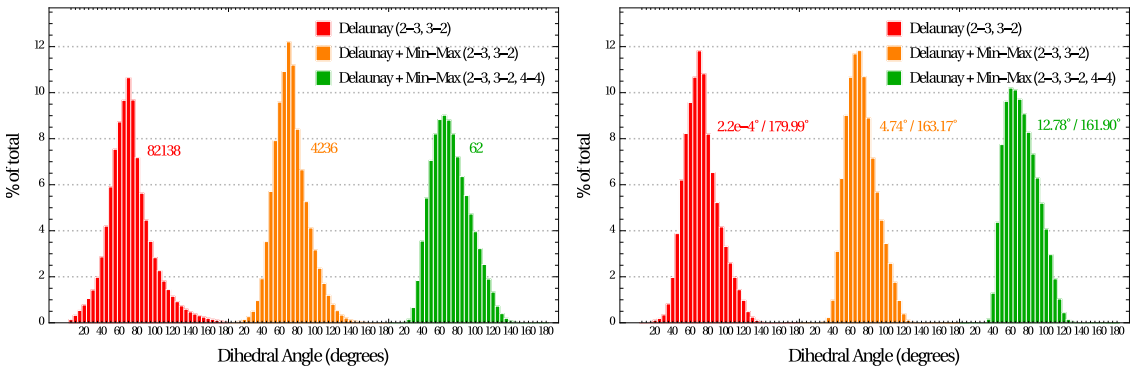


(a) Delaunay (2-3, 3-2) (b) Del. + Min-Max (2-3, 3-2) (c) Del. + Min-Max (2-3, 3-2, 4-4)

Figure 94: Slivers after the completion of refinement for varied quality criteria and transformations (Table 19). (a): 82138 slivers (0.097%) out of total 84.18 M tetrahedra; (b): 4236 slivers (0.0049%) out of total 86.24 M tetrahedra; (c): 62 slivers $(7.8\text{e-}5\%)$ out of total 78.97 M tetrahedra. Red signifies the slivers with a dihedral angle smaller than 2° or larger than 178°.

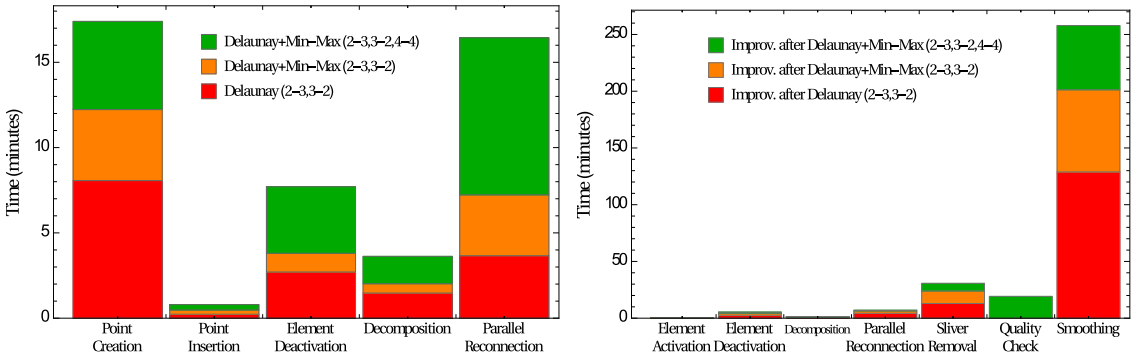
reason is the higher cost for grid improvement in the latter case. Figure 95 compares the quality histograms of the grids.

Figure 96 depicts a breakdown of the timing performance of the components at the refinement and the improvement step. Parallel reconnection with a combined criterion and combinations of 2-3, 3-2, and 4-4 flips takes 2.5 longer than reconnection with a single criterion (Figure 96(a)). Point creation takes a considerable amount of time, which depends on the number of grid generation passes. Smoothing time dominates the improvement step as it accounts for $85 - 90\%$ of the total improvement time (Figure 96(b)). On the other hand, the overheads due to reconnection are low because the number of active elements decreases rapidly.



(a) Before improvement (the number of slivers is reported). (b) After improvement (the min/max dihedral angles are reported).

Figure 95: Element angle distribution (in 5-deg increments) of Lv2b grids generated with varied quality criteria and transformations. Delaunay: In-sphere criterion; Min-Max: maximization of the minimum Laplacian edge weight. 2-3, 3-2, and 4-4 are flip types.

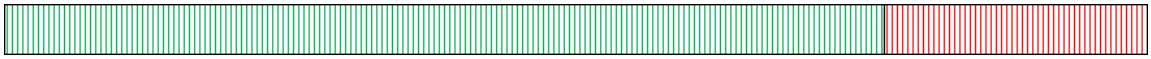


(a) Refinement step (b) Improvement step

Figure 96: Breakdown of the total CDT3D time for grid generation with varied quality criteria and transformations (Lv2b geometry). Delaunay: In-sphere criterion; Min-Max: maximization of the minimum Laplacian edge weight. 2-3, 3-2, and 4-4 are flip types. Parallel reconnection is performed using 12 cores. At refinement step a combined criterion with 2-3, 3-2, and 4-4 flips is slower than the other configurations. On the other hand, an improvement step that follows after refinement using a single criterion with 2-3 and 3-2 flips takes longer.

4.10.4 GROUPED OR SHUFFLED ELEMENT LINK-LIST?

CDT3D stores the elements in a double link-list data structure. The link-list contains both active and inactive elements. Some components (i.e., point creation, element deactivation, over-decomposition, and parallel reconnection) do not require a traversal of the inactive elements, therefore a separation between active and inactive elements can potentially improve the performance (Figure 97).



(a) Element link-list with grouped elements.

(b) Element link-list with shuffled elements.

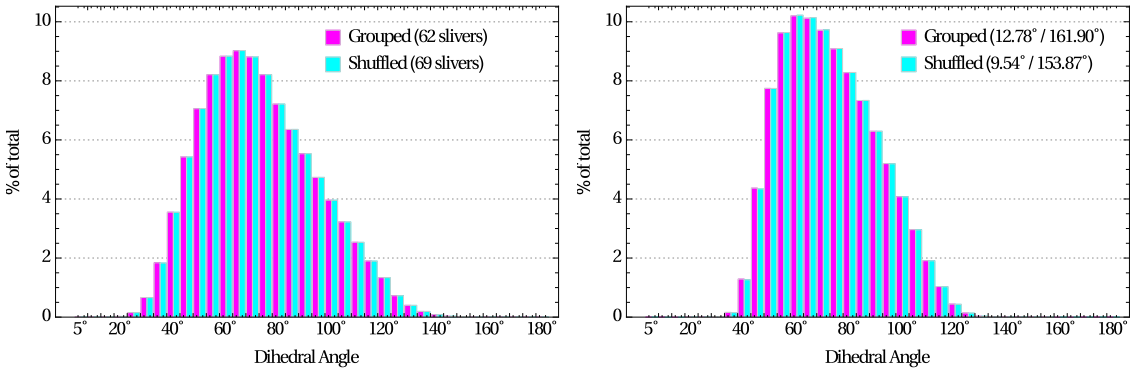
Figure 97: The two types of link-lists in CDT3D. Green represents the active elements. Red represents the inactive elements. A grouped link-list can be employed either for refinement or improvement using parameters $ordt$ and $ordtq$, respectively (Appendix B.3).

Table 20 reports performance results using the two types of link-lists. Figure 98 depicts the quality histogram before and after quality improvement. The element quality is slightly affected by the type of the link-list.

More important, a grouped link-list reduces threefold the refinement time compared to a shuffled link-list. Figure 99(a) illustrates the improvements in more detail. On the other hand, the timing performance in quality improvement is worse, when a grouped link-list is employed; however, this is because other operations (i.e., smoothing, quality check, and sliver removal) irrelevant with the type of the link-list, are take longer.

Table 20: Grid generation performance using a link-list with grouped and shuffled elements (Lv2b geometry). The grouped link-list is activated using parameters $ordt$ and $ordtq$, for refinement and improvement, respectively (Appendix B.3). The rest of the parameters are the same as in Table 19 (Delaunay + Min-Max (2-3, 3-2, 4-4) criterion). The experiments are performed on a DELL workstation with Linux Ubuntu 12.10, 12 hardware cores Intel(R) Xeon(R) CPU X5690@3.47 GHz, and 96 GB RAM.

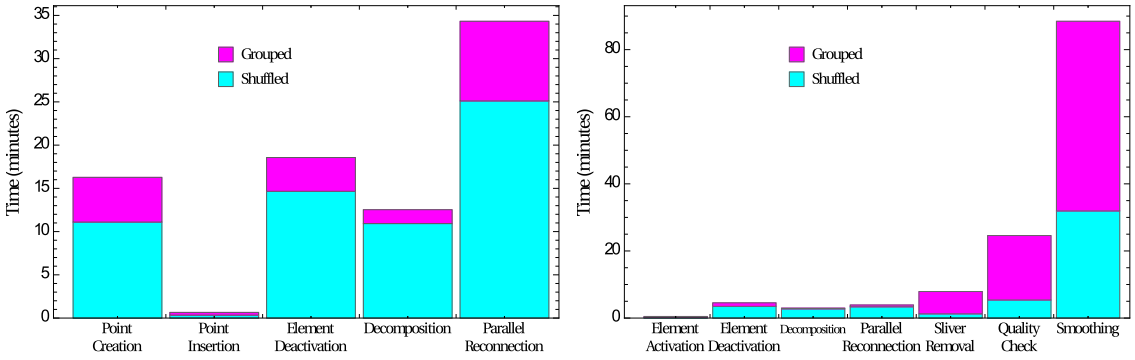
Link-list	Refinement			Improvement				Total
	#Tets (M)	#Slivers ($< 2°$ or $> 178°$)	Time (min)	#Tets (M)	#Slivers ($< 2°$ or $> 178°$)	Min/Max Dihedral Angle	Time (min)	Time (min)
Grouped	78.97	62	20.19	78.95	0	$12.78°/161.90°$	84.39	104.58
Shuffled	88.88	69	62.15	88.87	0	$9.54°/153.89°$	48.43	110.58



(a) Before improvement (the number of slivers is reported).

(b) After improvement (the min/max dihedral angles are reported).

Figure 98: Element angle distribution (in 5-deg increments) of Lv2b grids generated using a grouped and a shuffled element link-list.



(a) Refinement

(b) Improvement

Figure 99: Breakdown of the total CDT3D time for grid generation with a grouped and a shuffled element link-list. Parallel reconnection is performed with 12 hardware cores.

4.10.5 RESULTS WITH SPECULATIVE EXECUTION

A total of 61 runs are conducted to assess the performance of the CDT3D for varied granularities for over-decomposition, and fractions for bucket-migration. Each run is performed with 24 hardware cores. The granularity is adjusted with parameter $nbuckets$. The higher the number of buckets, the finer the decomposition. The fraction for bucket-migration is adjusted with parameter $frbtransf$. The higher the fraction, the higher the number of buckets to be transferred between threads. The rest of the parameters are fixed among the runs. The experiments are conducted on the Lv2b geomatry.

Figure 100 depicts the results. The element count ranges from 70M to 91M (Figure 100(a)). The tetrahedral grids are of a high quality. In the worst case scenario, only 0.00017% of the elements have a dihedral angle smaller than 2° or larger than 178° (without improvement) (Figure 100(b)). Decompositions with $nbuckets/24 > 20$ increase the number of grid generation passes (Figure 100(c)). Finer decompositions ($nbuckets/24 > 40$) raise the number of passes up to five times. The number of passes remains relatively the same when no data-migration is performed, regardless of the level of granularity.

The average reconnection time per iteration is computed to take into account the various number of grid generation passes. In general, finer decompositions result in faster reconnection (Figure 100(d)). A small improvement in running time occurs even in the absence of data-migration because repetitive reconnection completes faster in buckets with fewer elements.

The speedup is computed as the ratio of the time taken by one thread to the time taken by 24 threads. The parallel algorithm exhibits very good scalability. The best speedup obtained with 24 cores is 28.57 (Figure 100(f)). A super-linear speedup is achieved in some configurations. Nevertheless, the speedup of the parallel reconnection deteriorates on those configurations with a higher number of iterations (Figure 100(g)). In addition, the overal speedup is limited by the inverse of the fraction of sequential operations, such as point creation, point insertion, element deactivation, and decomposition (Figure 100(h)). Overall, the speculative results show that CDT3D achieves a good trade-off between the percentage of slivers, the number of grid generation iterations and the reconnection time, when $nbuckets/24 : 15 - 20$ and $frbtransf : 0.2 - 0.6$.

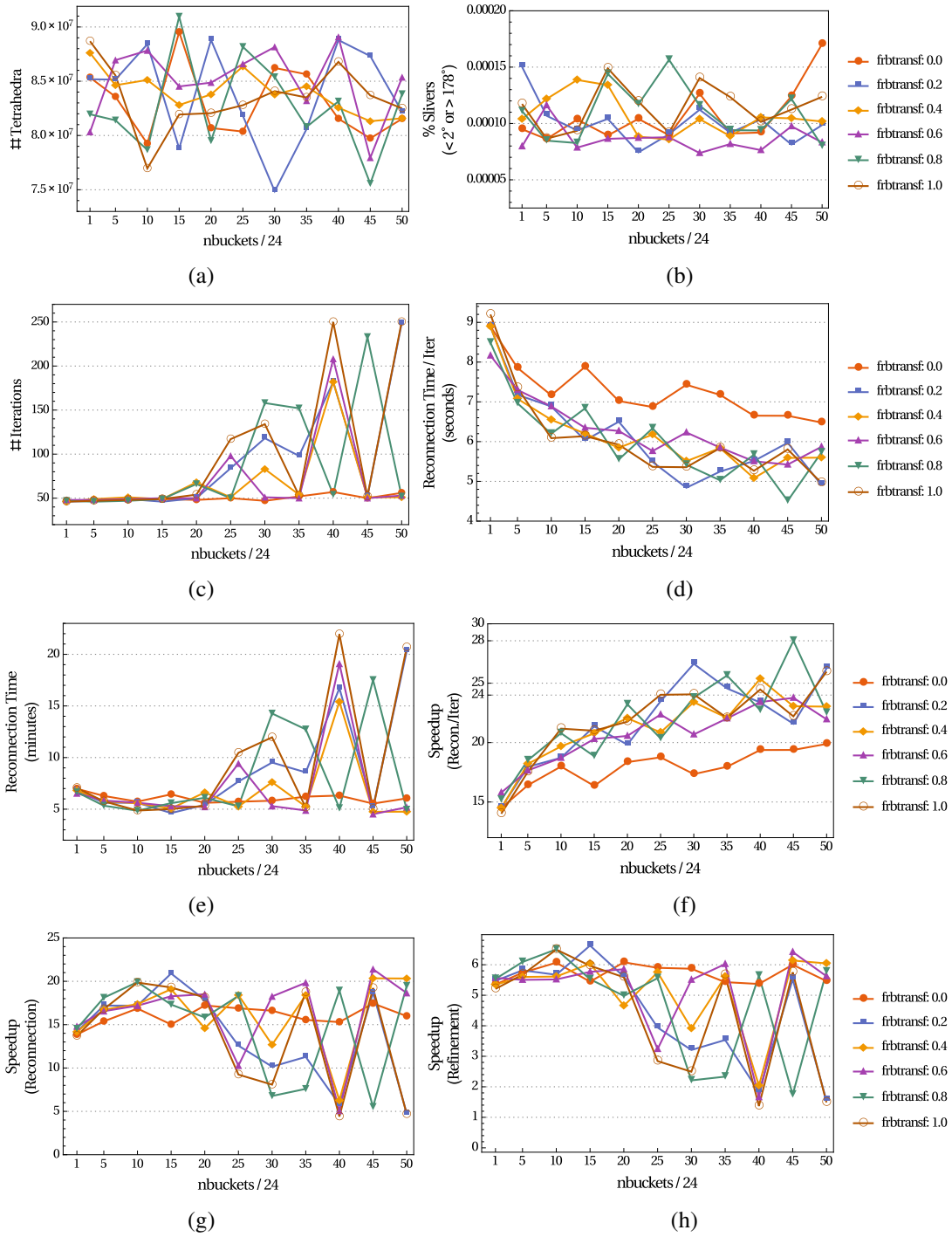


Figure 100: Performance results on parallel reconnection and refinement of Lv2b grid, for varied granularities for over-decomposition (nbuckets/24), and fractions for bucket-migration (frbtransf). Parameters: $nthreads$: 24; $cdfn$: 0.7; $cdfm$: 0.8 $mrecm2$; $mrec4$: 1; $nqual$: 0; $cbtransf$: 1. All runs performed using 24 threads on a DELL workstation with Linux Red Hat Enterprise, 24 hardware cores Intel(R) Xeon(R) CPU E5-2697v2@2.70 GHz, and 757 GB RAM.

4.10.6 RESULTS WITH HIGHER CONFIDENCE

This set of experiments performs a statistical evaluation on the performance of the grid generation. A total of 61 grids is generated for the nacelle geometry. The CDT3D parameters are fixed among the runs except the number of cores which varies from 1 to 24 in increments of 4. For each number of cores, the minimum, maximum, and median values of various results are reported, out of ten consecutive runs. The sequential run is performed once since the output is deterministic. Table 21 reports the results. Figure 101 illustrates the results using box-and-whisker together with the 25% and 75% quantiles.

Table 21: Grid generation results with higher confidence (aircraft nacelle geometry). For each number of cores, the minimum, maximum, and median values are reported out of ten consecutive runs. The sequential run is performed once. Parameters: $cdfn : 0.7$; $cdfm : 0.3$; $frbtransf : 0.4$; $cbtransf : 1$; $mrecm : 2$; $mrec4 : 0$; $nqual : 3$; $nsmth : 2$; $nbuckets : 15 \cdot nthreads$. Appendix B.3 lists the CDT3D parameters. The experiments performed on a DELL workstation with Linux Red Hat Enterprise, 24 hardware cores Intel(R) Xeon(R) CPU E5-2697v2@2.70 GHz, and 757 GB RAM.

		nthreads																			
		1	4			8			12			16			20			24			
			min	max	med	min	max	med	min	max	med	min	max	med	min	max	med	min	max	med
Other	#Tets (M)	54.08	46.71	55.48	50.84	42.53	52.86	47.64	46.82	55.63	51.32	42.58	55.53	49.92	47.34	56.21	51.94	41.37	60.32	51.60
	#Iterations	228	218	233	225	209	224	218	216	233	225	211	232	223	215	235	226	206	241	226
Quality	%Slivers($\times 10^{-2}$)	0.37	0.33	0.36	0.34	0.35	0.41	0.37	0.33	0.40	0.36	0.34	0.37	0.36	0.35	0.38	0.36	0.34	0.38	0.36
	#Slivers	2032	1643	1945	1769	1633	2034	1791	1685	2044	1883	1596	2068	1801	1687	2112	1900	1578	2224	1886
	Min Angle	11.28°	9e-06°	14.05°	9.17°	7.99°	15.87°	12.89°	0.001°	14.57°	11.21°	0.003°	14.74°	11.34°	0.51°	13.84°	8.89°	3.15°	14.71°	10.28°
	Max Angle	159.40°	149.52°	179.99°	160.61°	147.12°	159.57°	153.99°	151.63°	179.99°	158.56°	152.30°	179.99°	158.96°	151.26°	173.79°	159.19°	152.68°	170.38°	159.51°
Time	Recon./Iter (sec)	5.61	1.65	1.89	1.76	0.82	0.98	0.89	0.62	0.71	0.66	0.44	0.54	0.49	0.39	0.44	0.41	0.29	0.39	0.35
	Refinement (min)	25.81	9.99	12.10	10.98	6.36	8.13	7.29	6.10	7.55	6.86	5.07	6.70	6.02	5.20	6.36	5.88	4.39	6.74	5.60
	Smoothing (min)	17.95	14.04	93.10	30.03	12.91	26.24	17.57	14.23	70.34	23.05	12.95	60.91	24.40	12.82	51.56	22.83	12.23	46.60	27.75
	Improvement (min)	23.79	16.14	108.53	34.94	14.87	29.56	19.75	16.12	80.72	26.06	14.62	70.07	28.07	14.72	61.49	26.25	13.79	55.45	31.38
	Total (min)	49.63	26.78	120.11	45.95	22.06	36.82	27.06	22.92	87.31	32.94	20.27	76.10	34.12	20.47	67.42	32.15	19.05	61.24	37.01
Speedup	Recon./Iter	1	2.96	3.39	3.17	5.69	6.81	6.25	7.88	8.97	8.38	10.32	12.74	11.26	12.67	14.31	13.47	14.12	18.93	16.07
	Refinement	1	2.13	2.58	2.35	3.17	4.05	3.54	3.41	4.22	3.76	3.85	5.08	4.28	4.05	4.96	4.38	3.82	5.87	4.60
	Improvement	1	0.22	1.47	0.68	0.80	1.59	1.20	0.29	1.47	0.91	0.34	1.62	0.84	0.38	1.61	0.90	0.43	1.72	0.75
	Total	1	0.41	1.85	1.08	1.34	2.24	1.83	0.56	2.16	1.50	0.65	2.44	1.45	0.73	2.42	1.54	0.81	2.60	1.34

Parallelism influences all aspects of grid generation, such as quality before and after improvement, running time, number of refinement passes, and element count. For example, a minimum of 41.3 M tetrahedra and a maximum of 60.3 M tetrahedra are generated at ten consecutive runs with 24 cores (Figure 101(a)). This range is equal to 36% of the median value (51 M). For the same number of cores, the refinement concludes at a minimum of 206 passes and at a maximum of 241 passes (Figure 101(b)). This range is equal to 15.5% of the median value (226 passes).

Parallel reconnection affects the shape of the elements. Nevertheless, the number of generated slivers (without improvement) is very low (Figure 101(c)-101(d)). Note that the 4-4 flips are disabled in this set of experiments, hence a higher number of slivers should be expected. Additionally, the element quality after the improvement is high. The [min, max] values of the minimum and the maximum dihedral angle is $[8.89°, 12.89°]$ and $[153.99°, 160.61°]$, respectively (median values). Few sliver elements may survive

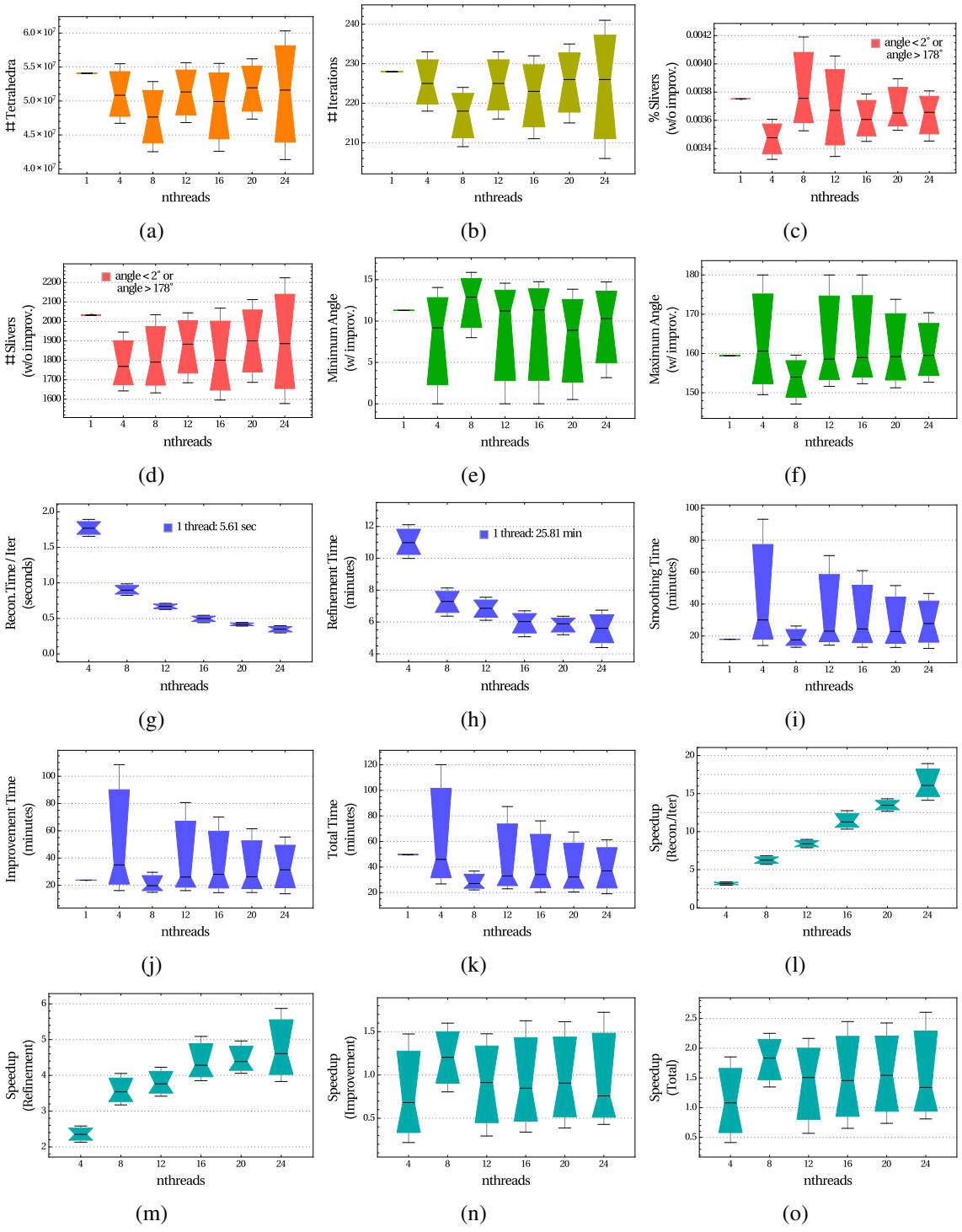


Figure 101: Box-and-whisker charts of the results in Table 21. For each number of threads, the minimum, maximum, median, 25% quantile, and 75% quantile values are depicted, out of ten consecutive runs. The sequential run is performed once. The experiments conducted on a DELL workstation with Linux Red Hat Enterprise, 24 hardware cores Intel(R) Xeon(R) CPU E5-2697v2@2.70 GHz, and 757 GB RAM.

(Figure 101(e)-101(f)). Those slivers can be eliminated by adjusting the parameters of the improvement algorithm (e.g., number of improvement passes). The improvement algorithm has a handful of options for quality grid generation and optimization, and these experiments are far from comprehensive. Appendix B.3 lists the CDT3D parameters.

Figure 101(g) illustrates the average reconnection time per iteration and Figure 101(l) depicts the corresponding speedup. The speedup is computed as the ratio of the time taken by one thread to the time taken by $nthreads$. Parallel reconnection exhibits good scalability. The minimum, maximum, and median speedup out of ten consecutive runs with 24 threads is 14.12, 18.93, and 16.07, respectively (Table 21). It should be understood, that the default combinations of 2-3 and 3-2 flips result in a lower speedup compared to combinations of 2-3, 3-2, and 4-4 flips (Figure 100(f)).

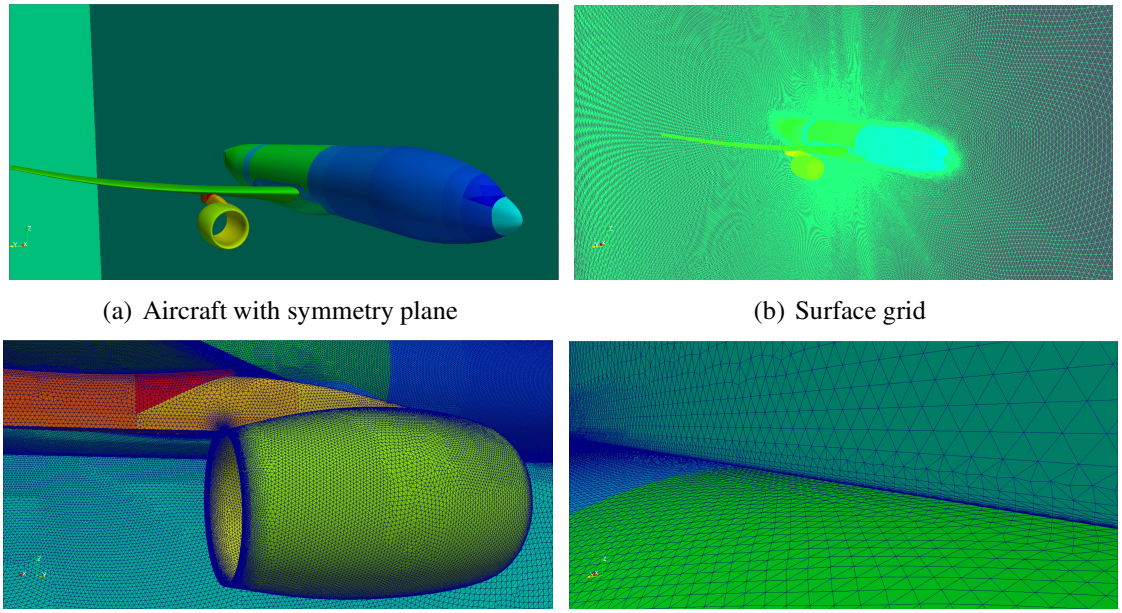
The scalability of the refinement algorithm is limited due to the cost of sequential operations, such as point creation, point insertion, element deactivation, and decomposition. The minimum, maximum, and median speedup for refinement with 24 threads is 3.82, 5.87, and 4.60, respectively (Figure 101(m)). The scalability of the improvement algorithm is very limited (Figure 101(n)) because smoothing accounts for up to 88.43% of the total improvement time (median value) (Table 21). Moreover, the number of eligible elements for reconnection decreases rapidly during quality improvement, therefore less concurrency can be achieved compared to the refinement step.

Figure 101(o) illustrates the speedup of the entire generation procedure, including an initial grid generation (i.e., Delaunay tetrahedralization and boundary recovery), a refinement, and an improvement step. The highest (median) speedup is obtained with 8 threads (1.83) mainly due to a lower cost for smoothing and a lower number of refinement iterations.

4.10.7 RESULTS USING NASA'S COMMON RESEARCH MODEL

This study evaluates the scalability of the CDT3D using NASA's Common Research Model[2]. The model is a DLR-F6 Airbus type aircraft. A tetrahedral grid of this model is generated with VGRID[3] [157, 158] for the purposes of the 6th AIAA CFD Drag Prediction Workshop[4]. After the surface of this grid is extracted using UGC[5], it is passed to CDT3D for volume grid generation. Figure 102 depicts the input surface grid.

[2]https://commonresearchmodel.larc.nasa.gov/2012/01/19/hello-world-2
[3]https://geolab.larc.nasa.gov/GridTool/Training/VGRID/
[4]https://aiaa-dpw.larc.nasa.gov
[5]http://www.simcenter.msstate.edu/



(a) Aircraft with symmetry plane (b) Surface grid

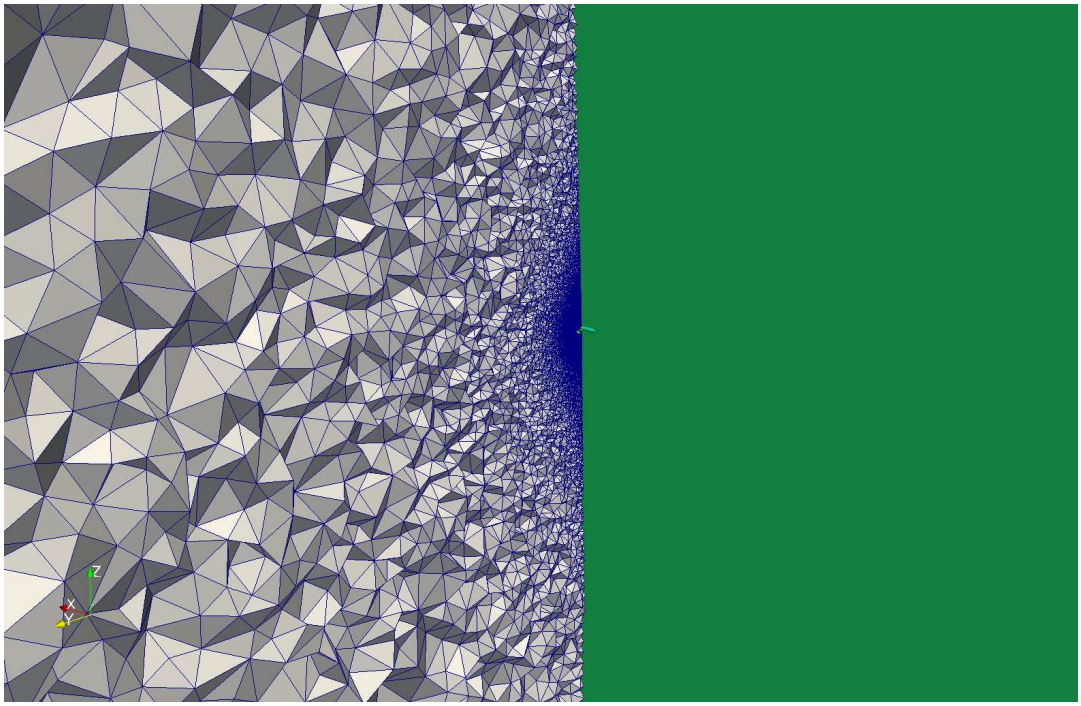
(c) Wing-nacelle-pylon system (d) Anisotropic boundary layers

Figure 102: Surface grid of a DLR-F6 Airbus type aircraft with anisotropic boundary layers on a symmetry plane; #points: 1006144; #triangles: 2012288.

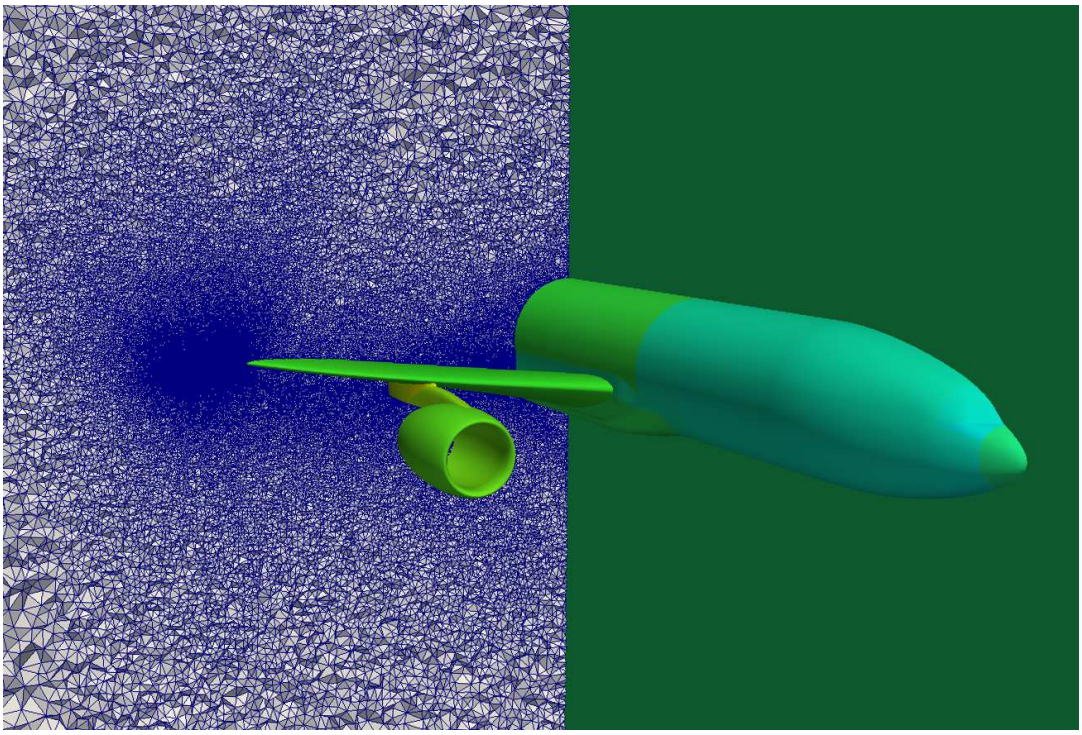
Boundary recovery is challenging, because the surface grid contains high aspect-ratio anisotropic triangles at the junction of the symmetry wall and the aircraft's body (Figure 102(d)). A tetrahedral grid is attempted to generate with AFLR, but the execution failed due to a topological error in boundary recovery.

This evaluation conducts a total of nine runs; a sequential run and two sets of parallel runs for varied number of threads (12-48). The first set of parallel runs is performed without element sorting (default option in CDT3D). The second set is performed with element sorting. Element sorting has been suggested in the literature [171] because it reduces the overlaps between regions involved in concurrent transformations, thus providing a good initial work load distribution for parallel reconnection. Element sorting is activated with parameter $sortp$ (Appendix B.3). The elements are sorted based on the coordinates of their centroids, according to a Hilbert curve [22, 179]. CDT3D sorts the elements in each refinement pass before over-decomposition.

Figure 103 depicts cuts of the isotropic tetrahedral grid. Table 22 presents the results. Parallel reconnection scales near-linearly with up to 24 threads. Element sorting results in a super-linear speedup when 12 and 24 threads are utilized. Speedup drops with a higher number of threads (Hyper-Threading is enabled).



(a) Overall view



(b) Detail view

Figure 103: Cuts of the tetrahedral grid of the flow field of DLR-F6 Airbus aircraft, generated with CDT3D. A smaller grid (≈200 M tetrahedra) is depicted due to limitations in visualization.

Table 22: Performance results on parallel refinement of grid of a flow domain around a DLR-F6 Airbus aircraft. Parameters: $cdfn : 0.8$; $cdfm : 0.5$; $mrecm : 2$; $mrec4 : 1$; $nbuckets : 15 \cdot nthreads$; $frbtransf : 0.4$; $cbtransf : 1.0$; $nqual : 0$; $sortp : 1$ activates element sorting. The included sorting time is reported in parenthesis. The sliver elements have a dihedral angle smaller than 2° or larger than 178°. #Iter is the number of grid generation passes. The experiments performed on a DELL workstation with Linux Red Hat Enterprise, 24 hardware cores Intel(R) Xeon(R) CPU E5-2697v2@2.70 GHz, and 757 GB RAM. Hyper-Threading is enabled when $nthreads > 24$.

			%Slivers		Time		SpeedUp	
	nthreads	#Tets (Bi)	(w/o improv.) ($\times 10^{-2}$)	#Iter	Recon./Iter (min)	Refinement (hours)	Recon./Iter	Refinement
	1	1.414	1.438	61	51.69	58.98	1	1
w/o sorting	12	1.413	1.472	73	4.81	13.10	10.74	4.50
	24	1.455	1.563	83	2.51	11.81	20.59	5.00
	36	1.438	1.487	79	1.98	10.59	26.10	5.57
	48	1.451	1.556	118	1.84	14.36	28.09	4.10
w/ sorting	12	1.414	1.563	89	3.99	17.38 (3.62)	12.95	3.39
	24	1.439	1.499	75	1.98	12.74 (3.16)	26.10	4.62
	36	1.458	1.518	93	1.67	14.87 (4.00)	30.95	3.96
	48	1.448	1.625	122	1.39	18.77 (5.08)	37.18	3.14

Figure 104 depicts plots of the results. The efficiency is computed as the ratio of the achieved speedup to the number of the threads. A program that scales linearly has a parallel efficiency of 1. Super-linear speedup leads to efficiency greater than 1. Table 22 reports the additional cost for element sorting in parenthesis. This cost is included in the refinement module. Also refinement includes overheads due to other sequential components. When 1 thread is utilized, the non-parallelized components together account for $(58.98 - (51.69 \cdot 61)/60)/58.98 = 10.89\%$ of the total refinement time. When 48 threads are utilized (without sorting), they account for 74.80%. When 48 threads are utilized and sorting is employed, the overheads due to sequential components increase to 84.94% (Table 22). Clearly, the end-to-end timing performance worsens when sorting is enabled.

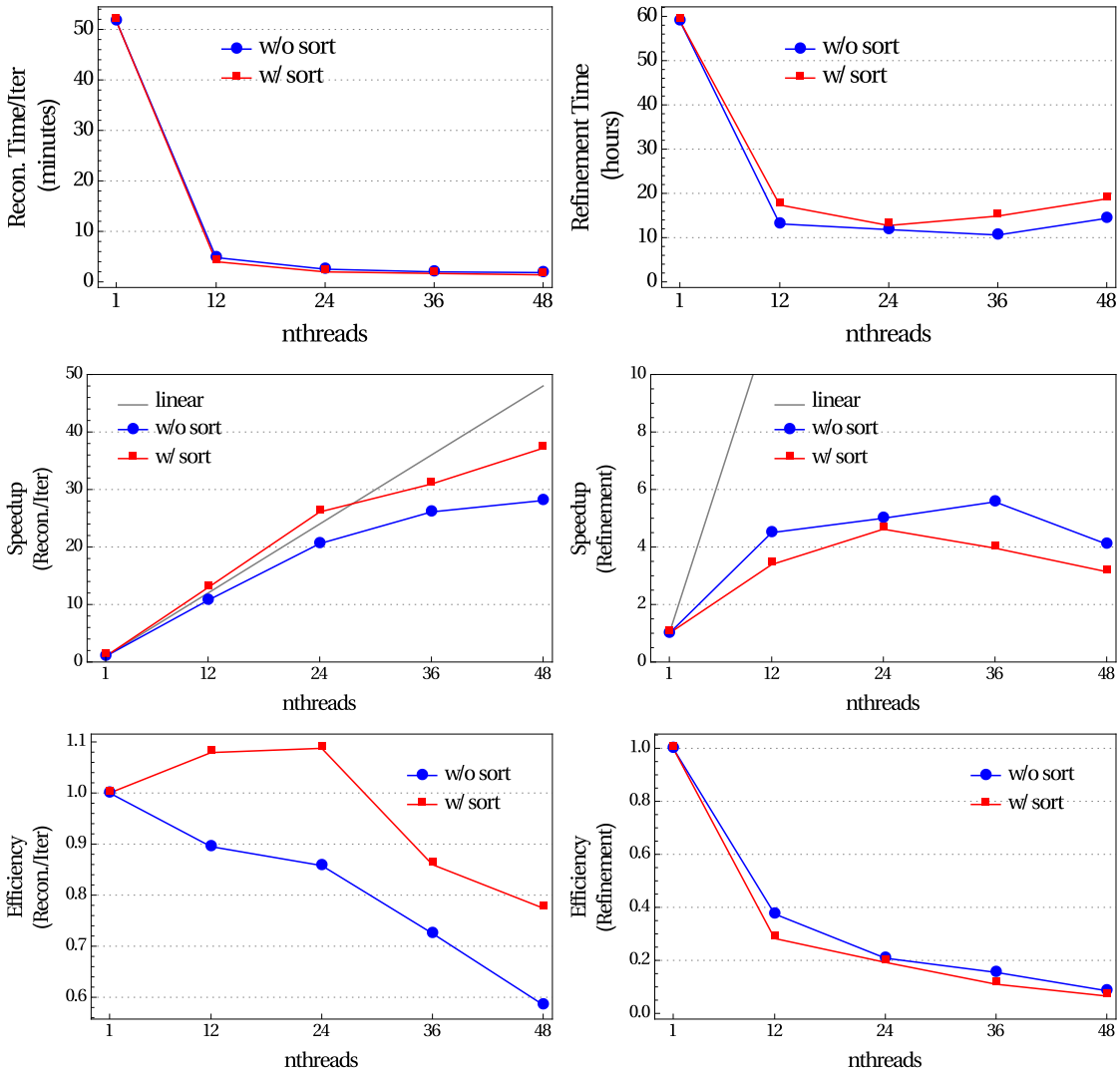


Figure 104: Scalability and efficiency of parallel reconnection and refinement of tetrahedral grid of DLR-F6 Airbus, with approximately 1.5 Billion elements (Table 22).

4.11 CONCLUSIONS

A new parallel optimistic unstructured grid generator for the discretization of piecewise linear complex domains with high quality isotropic elements has been presented. To the best of our knowledge, this is the first method that optimizes the grid connectivity in parallel throughout the generation procedure, including a post-improvement step. The connectivity optimization is based on a speculative approach that has been proven to perform well on hardware-shared memory (i.e., single chip). Our new speculative paradigm repetitively reconnects the grid using tightly-coupled topological transformations. Simple hill-climbing is employed to maximize the quality of the worst local element. The local reconnection scheme is based on: (i) data over-decomposition, (ii) atomic operations to avoid data races, and (iii) load-balancing to redistribute work-units among the threads. The grid generation engine is essentially a combination of Advancing Front type point placement, direct point insertion, and parallel multi-threaded connectivity optimization schemes.

The generator is evaluated on a variety of aerospace configurations. The results indicate that the high quality and performance attributes of this method see substantial improvement over existing state-of-the-art technology. The proposed framework will be part of an Extreme-Scale Anisotropic Mesh Generation Environment to meet industries expectations and NASA's CFD vision for 2030 [180, 43].

The presented unstructured grid generator fullfills the following requirements:

1. **Stability**. The grids generated (sequentially or parallel) by CDT3D are of comparable quality with the grids generated with state-of-the-art technology. Parallelism slightly affects the percentage of generated slivers before improvement and the final element quality after improvement.

2. **Robustness**. CDT3D exhibits "ultimate" robustness on recovering boundary constrained surfaces with high-aspect ratio or badly shaped triangles. CDT3D demonstrates excellent quality characteristics on a variety of realistic aerospace configurations. Isotropic tetrahedral grids with billions of elements are generated efficiently.

3. **End-User Productivity**. The sequential implementation of CDT3D refines the grid up to 2.5 times faster compared to state-of-the-art code. When 12 cores are utilized, CDT3D reduces the end-to-end grid generation time (including post-improvement)

up to a factor of 2, and the refinement time up to a factor of 9.8, compared to state-of-the-art technology. The parallel speculative connectivity optimization scheme exhibits near-linear scalability when 24 cores are utilized to discretize the flow domain of a DLR-F6 Airbus aircraft with approximately 1.5 billion elements.

4. **Code Re-Use**. The proposed software has a modular design. It includes four modules: (i) Delaunay Tetrahedralization, (ii) Boundary Recovery, (iii) Refinement, and (iv) Quality Improvement. Each module can be easily replaced and/or updated with a minimal effort.

4.12 FUTURE WORK

CDT3D is an end-to-end solution that focuses in both quality and performance aspects of mesh generation. In future efforts, this method could be further enhanced in the following ways:

1. Currently CDT3D interpolates a distribution function initially computed from the boundary surface. The distribution function determines the desired field point spacing. Other means of controlling the field point spacing, such as a geometric growth normal to boundary surfaces or a background grid may be also used.

2. Incorporate additional quality criteria for grid reconnection, such as a skewness metric [132]. Local reconnection using a Delaunay criterion followed by a Min-Max skew criterion typically results in a slightly improved quality at the expense of an increase in required CPU time.

3. Improve the scalability of the refinement module with the parallelization of the point creation, point insertion, element deactivation, and element sorting components.

4. Introduce more complex topological transformations $(n \geq 5)$ with pre-computation of the candidate solutions to potentially improve element quality [77]. Investigate the impact of those transformations in the performance and the convergence of the parallel optimistic connectivity optimization scheme.

5. Allow non-manifold surface connectivity throughout the generation process (e.g., boundary recovery, refinement, and improvement). Allow transparent source surfaces to be used as an input to control field point spacing nearby.

6. Incorporate *smart* smoothing techniques [71] in addition to optimal point placement and centroid averaging. Smoothing is a fundamental component of the improvement algorithm and highly influences the element quality. This experimental evaluation showed that vertex relocation is an expensive operation, as it takes about $80 - 90\%$ of the total improvement time, and should therefore be performed in parallel.

7. Improve the convergence of the parallel optimistic connectivity optimization.

8. Develop an Advancing-normal point type placement procedure to generate high aspect ratio tetrahedral, or mixed type (tetrahedral, prismatic, and hexahedral) elements [131]. A mixed element grid type could provide improved flow solver efficiency with the geometric flexibility of a fully unstructured tetrahedral grid. High aspect ratio elements are formed in layers near the boundary surface and are suitable for simulating viscous flow in computational fluid dynamics applications. Finally, a parallelization of the boundary layered algorithm should be considered.

REFERENCES

[1] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. *Journal of Algorithms*, 30(2):302–322, 1999.

[2] N. Amenta, S. Choi, and G. Rote. Incremental constructions con BRIO. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 211–219. ACM, 2003.

[3] L. Antiga, M. Piccinelli, L. Botti, B. Ene-Iordache, A. Remuzzi, and D. Steinman. An image-based modeling framework for patient-specific computational hemodynamics. *Medical & Biological Engineering & Computing*, 46(11):1097–1112, 2008.

[4] N. Archip, O. Clatz, S. Whalen, D. Kacher, A. Fedorov, A. Kot, N. Chrisochoides, F. Jolesz, A. Golby, P. M. Black, and S. K. Warfield. Non-rigid alignment of pre-operative MRI, fMRI, and DT-MRI with intra-operative MRI for enhanced visualization and navigation in image-guided neurosurgery. *NeuroImage*, 35(2):609 – 624, 2007.

[5] J. H. Argyris. Energy theorems and structural analysis: a generalized discourse with applications on energy principles of structural analysis including the effects of temperature and non-linear stress-strain relations part I. General theory. *Aircraft Engineering and Aerospace Technology*, 27(2):42–58, 1955.

[6] V. Arikatla, R. Ortiz, D. Thompson, D. Adams, A. Enquobahrie, and S. De. A hybrid approach to simulate tissue behavior during surgical simulation. In *4th International Conference on Computational and Mathematical Biomedical Engineering-CMBE2015*, 2015.

[7] M. A. Audette, K. Siddiqi, F. P. Ferrie, and T. M. Peters. An integrated range-sensing, segmentation and registration framework for the characterization of intra-surgical brain deformations in image-guided surgery. *Computer Vision and Image Understanding*, 89(2-3):226–251, 2003. Nonrigid Image Registration.

[8] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, 1976.

[9] S. Baden, N. Chrisochoides, D. Gannon, and M. Norman. *Structured Adaptive Mesh Refinement (SAMR) Grid Methods*. Springer-Verlag New York, 2000.

[10] T. J. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Engineering with Computers*, 5(3-4):161–175, 1989.

[11] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. PETSc 2.0 users manual. *Tech. Rep. ANL-95/11 - Revision 2.0.28, Argonne National Laboratory*, 2000.

[12] K. Barker, A. Chernikov, N. Chrisochoides, and K. Pingali. A load balancing framework for adaptive and asynchronous applications. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):183–192, 2004.

[13] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[14] T. BARTH. Numerical aspects of computing high reynolds number flows on unstructured meshes. In *29th Aerospace Sciences Meeting*, page 721, 1991.

[15] K.-J. Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[16] L. Bello, A. Castellano, E. Fava, G. Casaceli, M. Riva, G. Scotti, S. M. Gaini, and A. Falini. Intraoperative use of diffusion tensor imaging fiber tractography and subcortical mapping for resection of gliomas: technical considerations. *Neurosurgical Focus*, 28(2):E6, 2010.

[17] L. Bello, A. Gambini, A. Castellano, G. Carrabba, F. Acerbi, E. Fava, C. Giussani, M. Cadioli, V. Blasi, A. Casarotti, C. Papagno, A. K. Gupta, S. Gaini, G. Scotti, and A. Falini. Motor and language DTI fiber tracking combined with intraoperative subcortical mapping for surgical removal of gliomas. *NeuroImage*, 39(1):369–382, 2008.

[18] D. Benítez, E. Rodríguez, J. M. Escobar, and R. Montenegro. Performance evaluation of a parallel algorithm for simultaneous untangling and smoothing of tetrahedral meshes. In *Proceedings of the 22nd International Meshing Roundtable*, pages 579–598. Springer, 2014.

[19] P. Black. Brains, minds, and the surgical planning laboratory. In *SPL 25th Anniversary Reception, Brigham and Womens Hospital*, 20016.

[20] P. Blomstedt and M. Hariz. Hardware-related complications of deep brain stimulation: a ten year experience. *Acta neurochirurgica*, 147(10):1061–1064, 2005.

[21] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou. *Cilk: An efficient multithreaded runtime system*, volume 30. ACM, 1995.

[22] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 208–216. ACM, 2009.

[23] D. Boltcheva, M. Yvinec, and J.-D. Boissonnat. Mesh generation from 3D multi-material images. In G.-Z. Yang, D. Hawkes, D. Rueckert, A. Noble, and C. Taylor, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2009*, volume 5762 of *Lecture Notes in Computer Science*, pages 283–290. Springer Berlin Heidelberg, 2009.

[24] A. Bowyer. Computing dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[25] O. Brodersen. Drag prediction of engine-airframe interference effects using unstructured Navier-Stokes calculations. *Journal of Aircraft*, 39(6):927–935, 2002.

[26] J. Bronson, J. Levine, and R. Whitaker. Lattice cleaving: A multimaterial tetrahedral meshing algorithm with guarantees. *Visualization and Computer Graphics, IEEE Transactions on*, 20(2):223–237, Feb 2014.

[27] L. G. Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4):325–376, 1992.

[28] S. A. Canann, M. B. Stephenson, and T. Blacker. Optismoothing: An optimization-driven approach to mesh smoothing. *Finite Elements in analysis and Design*, 13(2-3):185–190, 1993.

[29] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[30] B. Chazelle. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing*, 13(3):488–507, 1984.

[31] B. Chazelle and L. Palios. Triangulating a nonconvex polytope. *Discrete & Computational Geometry*, 5(5):505–526, 1990.

[32] G. chen Sun, X. lei Chen, Y. Zhao, F. Wang, Z.-J. Song, Y. bo Wang, D. Wang, and B. nan Xu. Intraoperative MRI with integrated functional neuronavigation-guided resection of supratentorial cavernous malformations in eloquent brain areas. *Journal of Clinical Neuroscience*, 18(10):1350 – 1354, 2011.

[33] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Silver exudation. *Journal of the ACM (JACM)*, 47(5):883–904, 2000.

[34] A. N. Chernikov and N. P. Chrisochoides. Practical and efficient point insertion scheduling method for parallel guaranteed quality Delaunay refinement. In *Proceedings of the 18th annual international conference on Supercomputing*, pages 48–57. ACM, 2004.

[35] A. N. Chernikov and N. P. Chrisochoides. Parallel guaranteed quality Delaunay uniform mesh refinement. *SIAM Journal on Scientific Computing*, 28(5):1907–1926, 2006.

[36] A. N. Chernikov and N. P. Chrisochoides. Algorithm 872: Parallel 2D constrained Delaunay mesh generation. *ACM Transactions on Mathematical Software (TOMS)*, 34(1):6, 2008.

[37] A. N. Chernikov and N. P. Chrisochoides. Three-dimensional Delaunay refinement for multi-core processors. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 214–224. ACM, 2008.

[38] A. N. Chernikov and N. P. Chrisochoides. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM Journal on Scientific Computing*, 33(6):3491–3508, 2011.

[39] L. P. Chew. Guaranteed-quality Delaunay meshing in 3D (short version). In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, SCG '97, pages 391–393, New York, NY, USA, 1997. ACM.

[40] N. Chrisochoides. Parallel mesh generation. In *Numerical solution of partial differential equations on parallel computers*, pages 237–264. Springer, 2006.

[41] N. Chrisochoides, A. Fedorov, A. Kot, N. Archip, P. Black, O. Clatz, A. Golby, R. Kikinis, and S. K. Warfield. Toward real-time image guided neurosurgery using distributed and grid computing. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 37–37. IEEE, 2006.

[42] N. Chrisochoides and D. Nave. Parallel Delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering*, 58(2):161–176, 2003.

[43] N. P. Chrisochoides. Telescopic approach for extreme-scale parallel mesh generation for CFD applications. In *46th AIAA Fluid Dynamics Conference*, page 3181, 2016.

[44] N. P. Chrisochoides and F. Sukup. *Task parallel implementation of the Bowyer-Watson algorithm*, volume 235. Citeseer, 1996.

[45] O. Clatz, H. Delingette, I.-F. Talos, A. Golby, R. Kikinis, F. Jolesz, N. Ayache, and S. Warfield. Robust nonrigid registration to capture brain shift from intraoperative MRI. *Medical Imaging, IEEE Transactions on*, 24(11):1417–1427, Nov 2005.

[46] R. W. Clough. The finite element method in plane stress analysis. In *Second ASCE Conference on Electronic Computation*, pages 345–378, 1960.

[47] F. Commandeur, J. Velut, and O. Acosta. A VTK algorithm for the computation of the hausdorff distance. *The VTK Journal*, 2011.

[48] W. R. Crum, T. Hartkens, and D. Hill. Non-rigid image registration: theory and practice. *The British Journal of Radiology*, 2014.

[49] S. Curtis, R. Tamstorf, and D. Manocha. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, I3D '08, pages 61–69, New York, NY, USA, 2008. ACM.

[50] E. B. de Lisle and P. L. George. Optimization of tetrahedral meshes. In *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, pages 97–127. Springer, 1995.

[51] B. Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[52] O. Devillers, S. Pion, and M. Teillaud. Walking in a triangulation. *International Journal of Foundations of Computer Science*, 13(02):181–199, 2002.

[53] R. J. Dholakia, F. Drakopoulos, C. Sadasivan, X. Jiao, D. J. Fiorella, H. H. Woo, B. B. Lieber, and N. Chrisochoides. High fidelity image-to-mesh conversion for brain aneurysm/stent geometries. In *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, April 2015.

[54] R. J. Dholakia, A. Pagano, F. Drakopoulos, A. Kappel, C. Sadasivan, X. Jiao, D. J. Fiorella, N. Chrisochoides, H. H. Woo, and B. B. Lieber. In vitro and computational fluid dynamics comparison of the flow diversion efficacy of five commercial stents. In *Summer Biomechanics, Bioengineering and Biotransport Conference*, 2015.

[55] N. L. Dorward, O. Alberti, B. Velani, F. A. Gerritsen, W. F. J. Harkness, N. D. Kitchen, and D. G. T. Thomas. Postimaging brain distortion: magnitude, correlates, and impact on neuronavigation. *Journal of Neurosurgery*, 88(4):656–662, 1998.

[56] P. Doshi. Long-term surgical and hardware-related complications of deep brain stimulation. *Stereotactic and functional neurosurgery*, 89(2):89–95, 2011.

[57] F. Drakopoulos and N. P. Chrisochoides. Accurate and fast deformable medical image registration for brain tumor resection using image-guided neurosurgery. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 4(2):112–126, 2016.

[58] F. Drakopoulos, Y. Liu, P. Foteinos, and N. P. Chrisochoides. Towards a real time multi-tissue adaptive physics based non-rigid registration framework for brain tumor resection. *Frontiers in Neuroinformatics*, 8(11), 2014.

[59] F. Drakopoulos, R. Ortiz, A. Enquobahrie, D. Sasaki-Adams, and N. Chrisochoides. Tetrahedral image-to-mesh conversion software for anatomic modeling of arteriovenous malformations. *Procedia Engineering*, 124:278–290, 2015. 24th International Meshing Roundtable.

[60] F. Drakopoulos, C. Yao, Y. Liu, and N. Chrisochoides. An evaluation of adaptive biomechanical non-rigid registration for brain glioma resection using image-guided neurosurgery. In *Computational Biomechanics for Medicine*, pages 111–122. Springer, 2017.

[61] V. Dyedov, D. R. Einstein, X. Jiao, A. P. Kuprat, J. P. Carson, and F. del Pin. Variational generation of prismatic boundary-layer meshes for biomedical computing. *International Journal for Numerical Methods in Engineering*, 79(8):907–945, 2009.

[62] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15(3):223–241, 1996.

[63] H. Elhawary, H. Liu, P. Patel, I. Norton, L. Rigolo, X. Papademetris, N. Hata, and A. J. Golby. Intra-operative real-time querying of white matter tracts during frameless stereotactic neuronavigation. *Neurosurgery*, 68(2):506, 2011.

[64] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin. *SOFA: A Multi-Model Framework for Interactive Physical Simulation*, pages 283–321. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[65] A. Fedorov, N. Chrisochoides, R. Kikinis, and S. Warfield. Tetrahedral mesh generation for medical imaging. In *8th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2005)*, 2005.

[66] C. A. Felippa. Introduction to finite element methods. *Course Notes, Department of Aerospace Engineeing Sciences, University of Colorado at Boulder, available at http://www. colorado. edu/engineering/Aerospace/CAS/courses. d/IFEM. d*, 2004.

[67] M. Ferrant, A. Nabavi, B. Macq, P. Black, F. A. Jolesz, R. Kikinis, and S. K. Warfield. Serial registration of intraoperative MR images of the brain. *Medical Image Analysis*, 6(4):337 – 359, 2002.

[68] M. Ferrant, A. Nabavi, B. Macq, F. A. Jolesz, R. Kikinis, and S. K. Warfield. Registration of 3-D intraoperative MR images of the brain using a finite-element biomechanical model. *IEEE Transactions on Medical Imaging*, 20(12):1384–1397, 2001.

[69] D. A. Field. Laplacian smoothing and Delaunay triangulations. *International Journal for Numerical Methods in Biomedical Engineering*, 4(6):709–712, 1988.

[70] P. A. Foteinos and N. P. Chrisochoides. High quality real-time image-to-mesh conversion for finite element simulations. *Journal of Parallel and Distributed Computing*, 74(2):2123–2140, 2014.

[71] L. Freitag, M. Jones, and P. Plassmann. A parallel algorithm for mesh smoothing. *SIAM Journal on Scientific Computing*, 20(6):2023–2040, 1999.

[72] L. Freitag, P. Knupp, T. Munson, and S. Shontz. A comparison of optimization software for mesh shape-quality improvement problems. Technical report, Argonne National Lab., IL (US), 2002.

[73] L. A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40(21):3979–4002, 1997.

[74] I. Fried. Condition of finite element matrices generated from nonuniform meshes. *AIAA Journal*, 10(2):219–221, 1972.

[75] M. Fuchs, M. Wagner, and J. Kastner. Boundary element method volume conductor models for eeg source reconstruction. *Clinical Neurophysiology*, 112(8):1400–1407, 2001.

[76] R. R. Garlapati, G. R. Joldes, A. Wittek, J. Lam, N. Weisenfeld, A. Hans, S. K. Warfield, R. Kikinis, and K. Miller. Objective evaluation of accuracy of intraoperative neuroimage registration. In *Computational Biomechanics for Medicine*, pages 87–99. Springer, 2013.

[77] P.-L. George and H. Borouchaki. Back to edge flips in 3 dimensions. In *IMR*, pages 393–402, 2003.

[78] P. L. George, H. Borouchaki, and E. Saltel. 'Ultimate' robustness in meshing an arbitrary polyhedron. *International Journal for Numerical Methods in Engineering*, 58(7):1061–1089, 2003.

[79] P. L. George, F. Hecht, and É. Saltel. Automatic mesh generator with specified boundary. *Computer methods in applied mechanics and engineering*, 92(3):269–288, 1991.

[80] C. Giussani, F.-E. Roux, J. Ojemann, E. P. Sganzerla, D. Pirillo, and C. Papagno. Is preoperative functional magnetic resonance imaging reliable for language areas mapping in brain tumor surgery? Review of language functional magnetic resonance imaging and direct cortical stimulation correlation studies. *Neurosurgery*, 66(1):113–120, 2010.

[81] A. J. Golby. *Image-guided neurosurgery*. Academic Press, 2015.

[82] J. M. Gonzlez-Darder, P. Gonzlez-Lpez, F. Talamantes, V. Quilis, V. Corts, G. Garca-March, and P. Roldn. Multimodal navigation in the functional microsurgical resection of intrinsic brain tumors located in eloquent motor areas: role of tractography. *Neurosurgical Focus*, 28(2):E5, 2010.

[83] G. J. Gorman, J. Southern, P. E. Farrell, M. Piggott, G. Rokos, and P. H. Kelly. Hybrid OpenMP/MPI anisotropic mesh smoothing. *Procedia Computer Science*, 9:1513–1522, 2012.

[84] A. Goshtasby, L. Staib, C. Studholme, and D. Terzopoulos. Nonrigid image registration: guest editors introduction. *Computer vision and image understanding*, 89(2):109–113, 2003.

[85] W. Grimson, R. Kikinis, F. A. Jolesz, and P. Black. Image-guided surgery. *Scientific American*, 280(6):54–61, 1999.

[86] C. Hamani and A. M. Lozano. Hardware-related complications of deep brain stimulation: a review of the published literature. *Stereotactic and functional neurosurgery*, 84(5-6):248–251, 2006.

[87] P. Hastreiter, C. Rezk-Salama, G. Soza, M. Bauer, G. Greiner, R. Fahlbusch, O. Ganslandt, and C. Nimsky. Strategies for brain shift evaluation. *Medical Image Analysis*, 8(4):447–464, 2004.

[88] D. J. Hawkes, D. Barratt, J. M. Blackall, C. Chan, P. J. Edwards, K. Rhode, G. P. Penney, J. McClelland, and D. L. Hill. Tissue deformation and shape models in image-guided interventions: a discussion paper. *Medical Image Analysis*, 9(2):163–175, 2005.

[89] M. Holden. A review of geometric transformations for nonrigid body registration. *Medical Imaging, IEEE Transactions on*, 27(1):111–128, Jan 2008.

[90] M. Holden, J. A. Schnabel, and D. L. Hill. Quantification of small cerebral ventricular volume changes in treated growth hormone patients using nonrigid registration. *IEEE Transactions on Medical Imaging*, 21(10):1292–1301, 2002.

[91] X. Hu, X. Jiang, X. Zhou, J. Liang, L. Wang, Y. Cao, J. Liu, A. Jin, and P. Yang. Avoidance and management of surgical and hardware-related complications of deep brain stimulation. *Stereotactic and functional neurosurgery*, 88(5):296–303, 2010.

[92] A. Hberg, K. A. Kvistad, G. Unsgrd, and O. Haraldseth. Preoperative blood oxygen level-dependent functional Magnetic Resonance Imaging in patients with primary brain tumors: Clinical application and outcome. *Neurosurgery*, 54(4):902–915, 2004.

[93] F. Incekara, O. Olubiyi, A. Ozdemir, T. Lee, L. Rigolo, and A. Golby. The value of pre-and intraoperative adjuncts on the extent of resection of hemispheric low-grade gliomas: a retrospective analysis. *Journal of Neurological Surgery Part A: Central European Neurosurgery*, 77(02):079–087, 2016.

[94] Y. Ito, A. M. Shih, A. K. Erukala, B. K. Soni, A. Chernikov, N. P. Chrisochoides, and K. Nakahashi. Parallel unstructured mesh generation by an advancing front method. *Mathematics and Computers in Simulation*, 75(5):200–209, 2007.

[95] M. Jenkinson, M. Pechaud, and S. Smith. BET2: MR-based estimation of brain, skull and scalp surfaces. In *Eleventh Annual Meeting of the Organization for Human Brain Mapping*, 2005.

[96] S. Ji, X. Fan, D. W. Roberts, A. Hartov, T. J. Schaewe, D. A. Simon, and K. D. Paulsen. Brain shift compensation via intraoperative imaging and data assimilation. *CRC handbook of imaging in biological mechanics. CRC Press, Boca Raton*, pages 229–240, 2014.

[97] B. Joe. Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM Journal on Scientific Computing*, 16(6):1292–1307, 1995.

[98] H. Johnson, G. Harris, and K. Williams. Brainsfit: mutual information rigid registrations of whole-brain 3D images, using the Insight Toolkit. *Insight Journal*, pages 1–10, 2007.

[99] K. Kamada, T. Todo, Y. Masutani, S. Aoki, K. Ino, T. Takano, T. Kirino, N. Kawahara, and A. Morita. Combined use of tractography-integrated functional neuronavigation and direct fiber stimulation. *Journal of Neurosurgery*, 102(4):664–672, 2005.

[100] G. Karypis and V. Kumar. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*, 1998.

[101] A. Kazakidi, F. Drakopoulos, C. Sadasivan, N. Chrisochoides, J. Ekaterinaris, and B. B. Lieber. Simulation of blood flow diversion in cerebral aneurysms. In *European Congress on Computational Methods in Applied Sciences and Engineering*, June 2016.

[102] H. Kekhia, L. Rigolo, I. Norton, and A. J. Golby. Special surgical considerations for functional brain mapping. *Neurosurgery Clinics of North America*, 22(2):111–132, 2011.

[103] G. E. Keles, E. F. Chang, K. R. Lamborn, T. Tihan, C.-J. Chang, S. M. Chang, and M. S. Berger. Volumetric extent of resection and residual contrast enhancement on initial surgery as predictors of outcome in adult patients with hemispheric anaplastic astrocytoma. *Journal of Neurosurgery*, 105(1):34–40, 2006.

[104] K. S. Kenneth. Sparse matrix techniques. *Circuit Analysis, Simulation and Design*, 3(1), 1986.

[105] M. F. Khan, K. Mewes, R. E. Gross, and O. Škrinjar. Assessment of brain shift related to deep brain stimulation surgery. *Stereotactic and functional neurosurgery*, 86(1):44–53, 2008.

[106] J. Kim, T. Panitanarak, and S. M. Shontz. A multiobjective mesh optimization framework for mesh quality improvement and mesh untangling. *International Journal for Numerical Methods in Engineering*, 94(1):20–42, 2013.

[107] D. R. Kincaid, J. R. Respess, D. M. Young, and R. R. Grimes. Algorithm 586: ITPACK 2C: A FORTRAN package for solving large sparse linear systems by adaptive accelerated iterative methods. *ACM Transactions on Mathematical Software (TOMS)*, 8(3):302–322, 1982.

[108] A. Klein, J. Andersson, B. A. Ardekani, J. Ashburner, B. Avants, M.-C. Chiang, G. E. Christensen, D. L. Collins, J. Gee, P. Hellier, J. H. Song, M. Jenkinson, C. Lepage, D. Rueckert, P. Thompson, T. Vercauteren, R. P. Woods, J. J. Mann, and R. V. Parsey. Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration. *Neuroimage*, 46(3):786–802, 2009.

[109] B. M. Klingner and J. R. Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th international meshing roundtable*, pages 3–23. Springer, 2008.

[110] P. M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I-A framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering*, 48(3):401–420, 2000.

[111] R. Krishnan, A. Raabe, E. Hattingen, A. Szelnyi, H. Yahya, E. Hermann, M. Zimmermann, and V. Seifert. Functional Magnetic Resonance Imaging-integrated neuronavigation: Correlation between lesion-to-motor cortex distance and outcome. *Neurosurgery*, 55(4):904–915, 2004.

[112] F. Labelle and J. R. Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *ACM Transactions on Graphics (TOG)*, volume 26, page 57. ACM, 2007.

[113] M. J. Lang, J. J. Kelly, and G. R. Sutherland. A moveable 3-Tesla intraoperative magnetic resonance imaging system. *Neurosurgery*, 68:168–179, 2011.

[114] C. L. Lawson. Properties of n-dimensional triangulations. *Computer Aided Geometric Design*, 3(4):231–246, 1986.

[115] I. Lekht, N. Brauner, J. Bakhsheshian, K.-E. Chang, M. Gulati, M. S. Shiroishi, E. G. Grant, E. Christian, and G. Zada. Versatile utilization of real-time intraoperative contrast-enhanced ultrasound in cranial neurosurgery: technical note and retrospective case series. *Neurosurgical focus*, 40(3):E6, 2016.

[116] H. Lester and S. R. Arridge. A survey of hierarchical non-linear medical image registration. *Pattern Recognition*, 32(1):129 – 149, 1999.

[117] L. Linardakis and N. Chrisochoides. Graded Delaunay decoupling method for parallel guaranteed quality planar mesh generation. *SIAM Journal on Scientific Computing*, 30(4):1875–1891, 2008.

[118] Y. Liu, A. Fedorov, R. Kikinis, and N. Chrisochoides. Real-time non-rigid registration of medical images on a cooperative parallel architecture. In *Bioinformatics and Biomedicine, 2009. BIBM'09. IEEE International Conference on*, pages 401–404. IEEE, 2009.

[119] Y. Liu, P. Foteinos, A. Chernikov, and N. Chrisochoides. Mesh deformation-based multi-tissue mesh generation for brain images. *Engineering with Computers*, 28(4):305–318, 2012.

[120] Y. Liu, A. Kot, F. Drakopoulos, C. Yao, A. Fedorov, A. Enquobahrie, O. Clatz, and N. P. Chrisochoides. An ITK implementation of a physics-based non-rigid registration method for brain deformation in image-guided neurosurgery. *Frontiers in Neuroinformatics*, 8(33), 2014.

[121] Y. Liu and J. Snoeyink. A comparison of five implementations of 3D Delaunay tessellation. *Combinatorial and Computational Geometry*, 52:439–458, 2005.

[122] Y. Liu, C. Yao, F. Drakopoulos, J. Wu, L. Zhou, and N. Chrisochoides. A nonrigid registration method for correcting brain deformation induced by tumor resection. *Medical physics*, 41(10), 2014.

[123] S. H. Lo. Volume discretization into tetrahedra-II. 3D triangulation by advancing front approach. *Computers & Structures*, 39(5):501–511, 1991.

[124] R. Löhner. A parallel advancing front grid generation scheme. *International Journal for Numerical Methods in Engineering*, 51(6):663–678, 2001.

[125] R. Löhner. A 2nd generation parallel advancing front grid generator. In *Proceedings of the 21st international meshing roundtable*, pages 457–474. Springer, 2013.

[126] R. Löhner. Recent advances in parallel advancing front grid generation. *Archives of Computational Methods in Engineering*, 21(2):127–140, 2014.

[127] R. Löhner and P. Parikh. Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8(10):1135–1149, 1988.

[128] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[129] D. Louis, H. Ohgaki, O. Wiestler, W. Cavenee, P. Burger, A. Jouvet, B. Scheithauer, and P. Kleihues. The 2007 who classification of tumours of the central nervous system. *Acta Neuropathologica*, 114(2):97–109, 2007.

[130] K. E. Lunn, A. Hartov, F. E. Kennedy, M. I. Miga, D. W. Roberts, L. A. Platenik, and K. D. Paulsen. 3D ultrasound as sparse data for intraoperative brain deformation model. In *Medical Imaging 2001*, pages 326–332. International Society for Optics and Photonics, 2001.

[131] D. Marcum. Generation of unstructured grids for viscous flow applications. In *33rd Aerospace Sciences Meeting and Exhibit*, page 212, 1995.

[132] D. L. Marcum and N. P. Weatherill. Unstructured grid generation using iterative point insertion and local reconnection. *AIAA Journal*, 33(9):1619–1625, 1995.

[133] P. Markelj, D. Tomaevi, B. Likar, and F. Pernu. A review of 3D/2D registration methods for image-guided interventions. *Medical Image Analysis*, 16(3):642 – 661, 2012. Computer Assisted Interventions.

[134] C. R. Maurer, D. L. G. Hill, A. J. Martin, H. Liu, M. McCue, D. Rueckert, D. Lloret, W. A. Hall, R. E. Maxwell, D. J. Hawkes, and C. L. Truwit. Investigation of intraoperative brain deformation using a 1.5-T interventional MR system: preliminary results. *IEEE Transactions on Medical Imaging*, 17(5):817–825, 1998.

[135] C. R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[136] M. J. McGirt, D. Mukherjee, K. L. Chaichana, K. D. Than, J. D. Weingart, and A. Quinones-Hinojosa. Association of surgically acquired motor and language deficits on overall survival after resection of glioblastoma multiforme. *Neurosurgery*, 65(3):463–470, 2009.

[137] M. Miga, D. Roberts, F. Kennedy, L. Platenik, A. Hartov, K. Lunn, and K. Paulsen. Modeling of retraction and resection for intraoperative updating of images. *Neurosurgery*, 49(1):75–84; discussion 84–5, 2001.

[138] M. Miga, T. Sinha, D. Cash, R. Galloway, and R. Weil. Cortical surface registration for image-guided neurosurgery using laser-range scanning. *Medical Imaging, IEEE Transactions on*, 22(8):973–985, Aug 2003.

[139] M. I. Miga. Computational modeling for enhancing soft tissue image guided surgery: an application in neurosurgery. *Annals of biomedical engineering*, 44(1):128–138, 2016.

[140] G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation, and partition. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 683–692. ACM, 1995.

[141] S. Mohrehkesh, A. Fedorov, A. B. Vishwanatha, F. Drakopoulos, R. Kikinis, and N. Chrisochoides. Large scale cloud-based deformable registration for image guided therapy. In *Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016 IEEE First International Conference on*, pages 67–72. IEEE, 2016.

[142] A. V. Moiyadi and P. Shetty. Direct navigated 3D Ultrasound for resection of brain tumors: a useful tool for intraoperative image guidance. *Neurosurgical focus*, 40(3):E5, 2016.

[143] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *IMR*, pages 103–114, 2003.

[144] M. E. Mortenson. Geometric modeling. 1997.

[145] A. Mostayed, R. R. Garlapati, G. R. Joldes, A. Wittek, A. Roy, R. Kikinis, S. K. Warfield, and K. Miller. Biomechanical model as a registration tool for image-guided neurosurgery: evaluation against BSpline registration. *Annals of biomedical engineering*, 41(11):2409–2425, 2013.

[146] M. Murphy, D. M. Mount, and C. W. Gable. A point-placement strategy for conforming Delaunay tetrahedralization. *International Journal of Computational Geometry & Applications*, 11(06):669–682, 2001.

[147] D. Nave, N. Chrisochoides, and L. P. Chew. Guaranteed: quality parallel delaunay refinement for restricted polyhedral domains. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 135–144. ACM, 2002.

[148] C. Nimsky, O. Ganslandt, P. Hastreiter, and R. Fahlbusch. Intraoperative compensation for brain shift. *Surgical Neurology*, 56(6):357 – 364, 2001.

[149] C. Nimsky, O. Ganslandt, P. Hastreiter, R. Wang, T. Benner, A. G. Sorensen, and R. Fahlbusch. Intraoperative Diffusion-Tensor MR Imaging: Shifting of white matter tracts during neurosurgical procedures-initial experience 1. *Radiology*, 234(1):218–225, 2005.

[150] C. Nimsky, O. Ganslandt, P. Hastreiter, R. Wang, T. Benner, A. G. Sorensen, and R. Fahlbusch. Preoperative and intraoperative diffusion tensor imaging-based fiber tracking in glioma surgery. *Neurosurgery*, 56(1):130–138, 2005.

[151] C. Nimsky, O. Ganslandt, B. von Keller, J. Romstock, and R. Fahlbusch. Intraoperative high-field-strength MR Imaging: implementation and experience in 200 patients 1. *Radiology*, 233(1):67–78, 2004.

[152] T. Okusanya and J. Peraire. 3-D parallel unstructured mesh generation. In *Proc. Joint ASME/ASCE/SES Summer Meeting*, 1997.

[153] D. A. Orringer, D. R. Vago, and A. J. Golby. Clinical applications and future directions of functional MRI. In *Seminars in neurology*, volume 32, pages 466–475. Thieme Medical Publishers, 2012.

[154] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM transactions on Mathematical Software*, 8(1):43–71, 1982.

[155] P. Papadopoulos, E. Venkatapathy, D. Prabhu, M. P. Loomis, and D. Olynick. Current grid-generation strategies and future requirements in hypersonic vehicle design, analysis and testing. *Applied Mathematical Modelling*, 23(9):705–735, 1999.

[156] S. Perotto and L. Formaggia. *New Challenges in Grid Generation and Adaptivity for Scientific Computing*. Springer, 2015.

[157] S. Pirzadeh. Unstructured viscous grid generation by advancing-front method. Technical report, NASA, 1993.

[158] S. Pirzadeh. Unstructured viscous grid generation by the advancing-layers method. *AIAA Journal*, 32(8):1735–1737, 1994.

[159] J.-P. Pons, F. Ségonne, J.-D. Boissonnat, L. Rineau, M. Yvinec, and R. Keriven. High-quality consistent meshing of multi-label datasets. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 198–210. Springer, 2007.

[160] J. Radon. Mengen konvexer körper, die einen gemeinsamen punkt enthalten. *Mathematische Annalen*, 83(1):113–115, 1921.

[161] R. Radovitzky and M. Ortiz. Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-Delaunay triangulation. *Computer Methods in Applied Mechanics and Engineering*, 187(3-4):543–569, 2000.

[162] J.-F. Remacle, V. Bertrand, and C. Geuzaine. A two-level multithreaded Delaunay kernel. *Procedia Engineering*, 124:6–17, 2015.

[163] L. Rineau and M. Yvinec. A generic software design for Delaunay refinement meshing. *Computational Geometry*, 38(1-2):100–110, 2007.

[164] P. Risholm, E. Samset, I.-F. Talos, and W. Wells. A non-rigid registration framework that accommodates resection and retraction. In *Information Processing in Medical Imaging*, volume 21, pages 447–458, 2009.

[165] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Discrete & Computational Geometry*, 7(3):227–253, 1992.

[166] R. Said, N. Weatherill, K. Morgan, and N. Verhoeven. Distributed parallel Delaunay mesh generation. *Computer methods in applied mechanics and engineering*, 177(1-2):109–125, 1999.

[167] S. P. Sastry and S. M. Shontz. A parallel log-barrier method for mesh quality improvement and untangling. *Engineering with Computers*, 30(4):503–515, 2014.

[168] S. P. Sastry, S. M. Shontz, and S. A. Vavasis. A log-barrier method for mesh quality improvement and untangling. *Engineering with Computers*, 30(3):315–329, 2014.

[169] S. Sathornsumetee, J. N. Rich, and D. A. Reardon. Diagnosis and treatment of high-grade astrocytoma. *Neurologic clinics*, 25(4):1111–1139, 2007.

[170] E. Schönhardt. Über die zerlegung von dreieckspolyedern in tetraeder. *Mathematische Annalen*, 98(1):309–312, 1928.

[171] M. Shang, C. Zhu, J. Chen, Z. Xiao, and Y. Zheng. A parallel local reconnection approach for tetrahedral mesh improvement. *Procedia Engineering*, 163:289–301, 2016.

[172] E. G. Shaw, B. Berkey, S. W. Coons, D. Bullard, D. Brachman, J. C. Buckner, K. J. Stelzer, G. R. Barger, P. D. Brown, M. R. Gilbert, and M. Mehta. Recurrence following neurosurgeon-determined gross-total resection of adult supratentorial low-grade glioma: results of a prospective clinical trial. *Journal of Neurosurgery*, 109(5):835–841, 2008.

[173] J. Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley*, 73:12, 2002.

[174] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18(3):305–363, 1997.

[175] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95. ACM, 1998.

[176] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, pages 86–95, New York, NY, USA, 1998. ACM.

[177] J. R. Shewchuk. Constrained Delaunay tetrahedralizations and provably good boundary recovery. In *IMR*, pages 193–204. Citeseer, 2002.

[178] J. R. Shewchuk. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. *Unpublished manuscript*, 65, 2002.

[179] H. Si. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):11, 2015.

[180] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: a path to revolutionary computational aerosciences. 2014.

[181] S. M. Smith. Fast robust automated brain extraction. *Human Brain Mapping*, 17(3):143–155, 2002.

[182] A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *IEEE Transactions on Medical Imaging*, 32(7):1153–1190, 2013.

[183] C. Studholme, D. Hill, and D. Hawkes. An overlap invariant entropy measure of 3D medical image alignment. *Pattern Recognition*, 32(1):71–86, 1999.

[184] K. Sun, T. S. Pheiffer, A. L. Simpson, J. A. Weis, R. C. Thompson, and M. I. Miga. Near real-time computer assisted surgery for brain shift correction using biomechanical models. *IEEE journal of translational engineering in health and medicine*, 2:1–13, 2014.

[185] I. Talos and N. Archip. Volumetric non-rigid registration for MRI-guided brain tumor surgery. Technical report, Surgical Planning Laboratory, Department of Radiology, Brigham and Women's Hospital, Harvard Medical School, 2007.

[186] I.-F. Talos, L. ODonnell, C.-F. Westin, S. Warfield, I. Wells, William, S.-S. Yoo, L. Panych, A. Golby, H. Mamata, S. Maier, P. Ratiu, C. Guttmann, P. Black, F. Jolesz, and R. Kikinis. Diffusion tensor and functional MRI fusion with anatomical MRI for image-guided neurosurgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 407–415. Springer, 2003.

[187] M. Tang, D. Manocha, and R. Tong. Fast continuous collision detection using deforming non-penetration filters. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 7–13, New York, NY, USA, 2010. ACM.

[188] M. Tang, S.-E. Yoon, and D. Manocha. Adjacency-based culling for continuous collision detection. *The Visual Computer*, 24(7):545–553, 2008.

[189] C. Tempany, J. Jayender, T. Kapur, R. Bueno, A. Golby, N. Agar, and F. A. Jolesz. Multimodal imaging for improved diagnosis and treatment of cancers. *Cancer*, 121(6):817–827, 2015.

[190] J. F. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of grid generation*. CRC press, 1998.

[191] P. Thompson and A. W. Toga. A surface-based technique for warping three-dimensional images of the brain. *IEEE Transactions on Medical Imaging*, 15(4):402–417, 1996.

[192] M. Turner, L. Topp, H. Martin, and R. Clough. Stiffness and deflection analysis of complex structures. *Journal of the Aeronautical Sciences*, 23:805–823, 1956.

[193] P. Van den Munckhof, M. F. Contarino, L. J. Bour, J. D. Speelman, R. M. A. de Bie, and P. R. Schuurman. The first evaluation of brain shift during functional neurosurgery by deformation field analysis. *Neurosurgery*, 67:49–54, 2010.

[194] L. M. Vigneron, S. K. Warfield, P. A. Robe, and J. G. Verly. 3D XFEM-based modeling of retraction for preoperative image update. *Computer Aided Surgery*, 16(3):121–134, 2011.

[195] D. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.

[196] N. P. Weatherill and O. Hassan. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37(12):2005–2039, 1994.

[197] M. R. Wiegell, H. B. Larsson, and V. J. Wedeen. Fiber crossing in human brain depicted with diffusion tensor MR imaging. *Radiology*, 217(3):897–903, 2000.

[198] D. Winkler, M. Tittgemeyer, J. Schwarz, C. Preul, K. Strecker, and J. Meixensberger. The first evaluation of brain shift during functional neurosurgery by deformation field analysis. *Journal of Neurology, Neurosurgery, and Psychiatry*, 76(8):1161–1163, 2005.

[199] A. Wittek, K. Miller, R. Kikinis, and S. K. Warfield. Patient-specific model of brain deformation: Application to medical image registration. *Journal of Biomechanics*, 40(4):919 – 929, 2007.

[200] J.-S. Wu, L.-F. Zhou, W.-J. Tang, Y. Mao, J. Hu, Y.-Y. Song, X.-N. Hong, and G.-H. Du. Clinical evaluation and follow-up outcome of diffusion tensor imaging-based functional neuronavigation: a prospective, controlled study in patients with gliomas involving pyramidal tracts. *Neurosurgery*, 61(5):935–949, 2007.

[201] R. Zangeneh. Thread-parallel mesh generation and improvement using face-edge swapping and vertex insertion. Master's thesis, University of British Columbia, 2014.

[202] Y. Zhang, C. Bajaj, and B.-S. Sohn. 3D finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, 194(48):5083–5106, 2005.

[203] Y. Zhang and J. Qian. Resolving topology ambiguity for multiple-material domains. *Computer Methods in Applied Mechanics and Engineering*, 247:166–178, 2012.

APPENDIX A

BIOMECHANICAL DEFORMABLE REGISTRATION FOR DEEP BRAIN STIMULATION

A.1 INTRODUCTION

In this paper we present an adaptive deformable registration method for Deep Brain Stimulation. The method relies on hexahedral mesh generation to compensate for the noise of CT scans. An adaptive approach is utilized to improve the accuracy of a well known non-rigid registration method used extensively for registration of MRI data. Finally, parallel computing is used to reduce the execution time, which increases with adaptivity. Our evaluation on three DBS cases indicates that the proposed scheme satisfies the real-time constraints of DBS surgery and recovers the deep-brain deformation with high fidelity. Understanding brain shift in this context is an important task to improve the patient outcomes in DBS surgery.

Deep brain stimulation (DBS) is an effective palliative therapy for patients suffering from Essential Tremor, Parkinson's disease, and other neurological movement disorders. As an adjunct to medical intervention, DBS therapy can reduce the morbidity associated with these disorders significantly [105]. DBS surgery involves the placement of electrical leads into precise locations in the deep structures of the brain without direct intraoperative visualization of the target structures or of the electrode lead (Figure 105). Modern DBS surgery makes use of stereotactic systems and image guidance to accurately place electrode leads, as well as intraoperative imaging to surveil the location of the lead and guide the surgery. The effectiveness of DBS is directly correlated with the accuracy of DBS electrode lead placement, with more accurate electrode placement leading to better clinical outcomes. Re-operation is rarely necessary to correct the placement of a DBS lead (1% to 12.7%)[20, 56, 91, 86]. The targets of the DBS surgery are located in the basal ganglia, a structure that helps regulate movement and is central to the pathology of Parkinson's disease. The basal ganglia itself includes the subthalamic nucleus (STN) and globus pallidus internus (GPi), two targets of DBS surgery [193, 198].

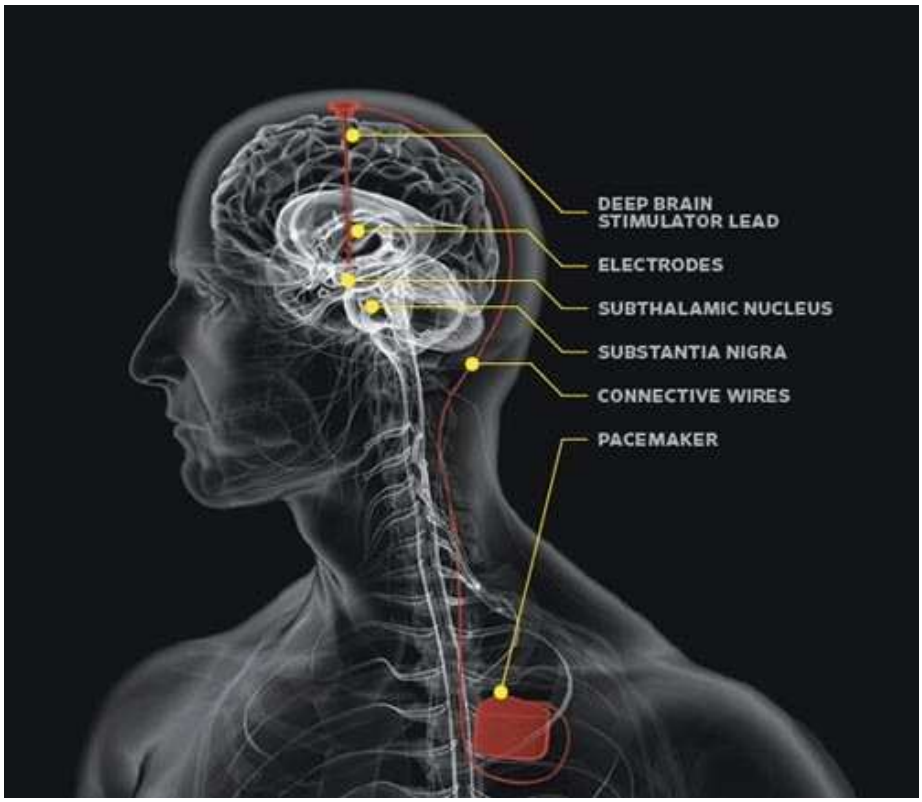


Figure 105: Deep Brain Stimulation (DBS). An implanted pacemaker delivers electrical stimulation to targeted areas in the brain that control movement, blocking the abnormal nerve signals that cause Tremor and Parkinsons disease symptoms. The image obtained from the Department of Neurosurgery at the University of Texas Health Science Center at San Antonio (UTHSCSA).

Compounding the relatively small size and deep location of these nuclei is the fact that they are often moving targets during surgery. The phenomenon of brain shift is well documented during DBS and other procedures that result in CSF leakage. Previous studies have shown a brain shift during DBS surgery of up to 4mm in deep brain structures, and up to 13mm in cortical structures, as well as development of a pneumocephalus in early post-surgery of up to 20mm. In addition to intraoperative shift, there is evidence that as subdural air collections resolve in the weeks after surgery, these leads may continue to shift [193, 198]. This shift complicates the placement of the DBS electrode leads because the target nuclei are not visible on intraoperative CT, the modality commonly available in community and academic medical centers. The accurate modeling and correction for intraoperative brain shift during DBS surgery is essential for the improvement of surgical outcomes.

In this paper we propose a method to detect and quantify brain shift during DBS surgery using preoperative (CT, MRI) and intraoperative (O-arm CT) imaging as inputs to a patient-specific biomechanical adaptive deformable (or non-rigid) registration algorithm [58]. In essence, this approach compares preoperative and intraoperative images, detecting their movement, and inferring the unknown position of the surgical targets after brain shift using the deformation recovered from these landmarks.

In this work, we describe a method to detect the brain shift that occurs during DBS surgery both qualitatively and quantitatively. The average brain shift of 2mm in the area of interest is at the boundary of resolution for this biomechanical deformable registration method; however, the presence of a highly radiolucent flexible intracranial fiducial (the stimulating electrode) improves the accuracy of this method. This effect is greatest precisely where it is most needed; near the deep-brain targets of DBS surgery. We use this fiducial as a gross landmark in the registration process that allows us to recover the detailed deformation of the electrode and the surrounding solid tissues. Finally, we employ recent advancements in computational power and the availability of powerful multi-processing GPU hardware to make the intraoperative use of these algorithms feasible and cost-effective. This high-performance computing architecture is widely available and a good candidate for intraoperative use.

A.2 PATIENTS AND METHODS

Three patients were included in this study. All patients had Parkinson's disease that was refractory to medical management and were treated with DBS surgery. All patients were treated at VCUHS Medical Center hospital or the VA McGuire Medical Center in Richmond. Patients were selected for this retrospective study based on several criteria: (i) amount of intracranial postoperative air, (ii) presence of bilateral air, (iii) amount of brain shift, and (iv) availability of preoperative images (MRI, CT), intraoperative images (O-arm CT) at appropriate procedural intervals, and post-operative images (CT). The image size and spacing of the acquired clinical O-arm CT data is: $512 \times 512 \times 192$ (voxels^3), and $0.415 \times 0.415 \times 0.833$ (mm^3), respectively.

A.2.1 PROCEDURAL METHODS

Pre-operative images for all patients were obtained, including fine-cut cone-beam CT, volumetric T1, T2, and FLAIR MRI protocols. Additionally, the placement of six bony fiducials in the patient's scalp assists in later rigid registrations. A stereotactic plan was

created using these images. The NexFrame stereotactic system by Medtronic, Inc. was used for all cases. Next, the stereotactic space was registered to the image space using the previously placed bony fiducials while optimizing registration accuracy. The patient's skull was trepanned and the dura opened to allow visualization of the brain. The stereotactic tower was aligned to target and then the rigid cannulas were inserted into the brain along the planned track trajectory. The patient was awoken and Microelectrode recording (MER) was then carried out to physiologically identify the target nucleus with the intent of correcting any targeting inaccuracy. An O-arm image was then obtained with the cannula and microelectrode at the target. Additional parallel tracks were made if necessary. Once the ideal track was determined, the microelectrode was then withdrawn, and the DBS lead inserted.

The same procedure is then repeated on the contralateral side. O-arm CT images are taken after insertion of the microcannula on the second side, and then again after final placement of both DBS leads. Following closure, the fiducials are removed and the patient is then taken for a second post-operative fine-cut CT scan.

A.2.2 COMPUTATIONAL METHODS

An Adaptive Physics-Based Non-Rigid Registration method (A-PBNRR) [57] customized to register 3D preoperative and intraoperative images for DBS is proposed. Given preoperative MRI and CT, as well as intraoperative MR (iMR) or CT (iCT), we aim to find a deformation field between them and then deform the preoperative MRI according to this field. The main idea of the physics-based non-rigid registration method is to use the known displacement vector associated with sparse feature points in the brain to estimate the entire brain deformation using a regularization term based on a brain biomechanical model [45]. This method includes four critical components: (i) Segmentation and Mesh Generation; generate a patient-specific model, (ii) Feature point detection; identify small image blocks that have rich structural information in the preoperative MRI, (iii) Block matching; calculate displacement for each image block to generate a sparse deformation field, and the main computational step (iv) Finite Element Solver; estimate entire brain deformation based on the sparse deformation field computed at the block matching step.

Instead of using a tetrahedral mesh, we use a hexaderal mesh, which is more robust for CT images with higher noise than MRI. Given the noise in CT scans and the fact that the relevance of a displacement estimated with a block matching algorithm depends on the existence of highly discriminative structures within a block, we briefly focus on

feature selection and block matching. We use the variance of the image intensity within the block region to measure its relevance, only selecting a fraction of all potential blocks based on a predefined parameter of the algorithm. To avoid redundancy by the overlapping of blocks (i.e., eliminate blocks that are too close to each other), a parameter of prohibited connectivity is used. There are three options for the connectivity: vertex, edge, and face connectivity (Figure 106). In addition to various connectivity patterns supported in the ITK implementation, we use face connectivity since it allows us to leverage the high confidence landmarks from the lead. Block matching is a well-known technique widely used in motion coding, image processing, and compression. It is based on the assumption that a complex non-rigid transformation can be approximated by point-wise translations of small image regions. Considering an image block in a floating image and a predefined search window in a reference image, the block matching algorithm searches for a position in the reference image that maximizes a similarity measure. Similarity measures in this task include mean square difference of intensity (MSD), mutual information (MI), and normalized cross correlation (NCC) [27]. By assembling the individual displacement vectors, one can create a sparse displacement field, which the finite element solver will use to approximate the unknown displacement vector associated with the mesh vertices.

A.2.3 ANALYSIS

The patient specific biomechanical non-rigid registration algorithm described above estimates the deformation that occurs between any two images. In particular, the registration algorithm accurately tracks highly radiopaque structures, such as the flexible electrode lead. Our hypothesis is that the deformation in the flexible electrode approximates the deformation in the deformable soft tissue (which is radiolucent). The deformations produced between successive intraoperative images is first visualized, and then qualitatively evaluated. We describe the deformation in the region of interest near to the deep basal ganglia nuclei, or in a diameter of 1cm around the flexible electrode lead. Next, the recovered deformation fields are used to deform the preoperative scan. Using this newly updated image, we measure the locations of the relevant nuclei and characterize the brain shift that accounts for this movement.

A.3 RESULTS

For each patient, we performed a deformable registration between O-arm CT1 and O-arm CT2 images. Table 23 lists the parameters of the registration. We identified the

preoperative and intraoperative positions of the relevant nuclei, as well as the deformation at the tip of the flexible lead from the deformed registered O-arm CT1. For the initialization of the non-rigid registration, we performed a rigid alignment using Slicer's 4.4.0 BRAINSFit module [98]. Table 24 presents some quantitative results of the deformable registration. In all cases, the deformation is for the first side lead. In cases 1-2, the tip of the lead shifted primarily posteriorly and toward the side of the lead. In case 3, the lead shifted anteriorly and toward the side of the lead. In our future work, we will include additional quantitative data regarding the shift in clinically important nuclei like the STN and the GPi in the MRIs, as well as additional measures of registration accuracy like the Hausdorff Distance. Figure 107 presents the qualitative evaluation results for case 1. The image discrepancies after our non-rigid registration are smaller compared to the rigid registration, particularly near the flexible electrode lead. Figure 108 depicts in more detail the recovered deformations for case 2. Figure 109 depicts qualitative evaluation results for case 3.

Table 23: The input parameters for the deformable registration (x: axial; y: coronal; z: sagittal).

Parameter	Units	Value	Description
Element type	-	8-node Hex	-
Similarity metric	-	NCC	Normalized Cross Correlation
F_s	-	5%	$\%$ selected image blocks
Connectivity pattern	-	"face"	-
$B_{s,x} \times B_{s,y} \times B_{s,z}$	voxels	$3 \times 3 \times 3$	Block size
$W_{s,x} \times W_{s,y} \times W_{s,z}$	voxels	$9 \times 9 \times 9$	Window size
$H_{s,x} \times H_{s,y} \times H_{s,z}$	-	$18 \times 18 \times 13$	Num Hexahedrons
E_b	Pa	2.1×10^3	Brain's Young modulus
ν_b	-	0.45	Brain's Poisson ratio
F_r	-	25%	% of rejected block outliers
N_{appr}	-	5	Num of outlier rejection steps
N_{int}	-	5	Num of interpolation steps
N_{iter}	-	1	Num of adaptive iterations

A.4 CONCLUSIONS

In this study, we measured the brain shift that occurs during deep brain stimulation surgery by using flexible DBS leads as a radiopaque landmark in a biomechanical deformable registration method. We show how it is possible to recover the deformations that

Table 24: Deformation (mm) at the tip of the flexible lead, end-to-end non-rigid registration time (including I/O) (seconds), and speed-up for the three clinical cases. The experiments conducted in a Linux workstation with 8 Intel i7-2600 @3.400 GHz CPU cores, and 16GB of RAM.

Case	Deformation (mm)		Time (sec)		Speed-Up
	Lead tip	Lead max	1 thread	8 threads	
1	1.05	2.21	233.07	100.70	2.31
2	1.26	2.08	212.76	90.10	2.36
3	1.57	3.18	219.99	94.01	2.34

affect two clinically important nuclei of the basal ganglia. This technique can be applied intraoperatively to improve the targeting of DBS leads, and post-operatively to understand the nature of shift during DBS surgery. Future work includes characterizing how the brain shift occurs over time using multiple intraoperative images acquired during a single surgery. In addition, using pre- and post-operative MRI, it becomes possible to recover the entire time course of shift, including any that occurs before the first intraoperative image is captured. A better understanding of this shift and the intraoperative correction of the same can, with future enhancements of this method, improve targeting accuracy and patient outcomes in DBS surgery.

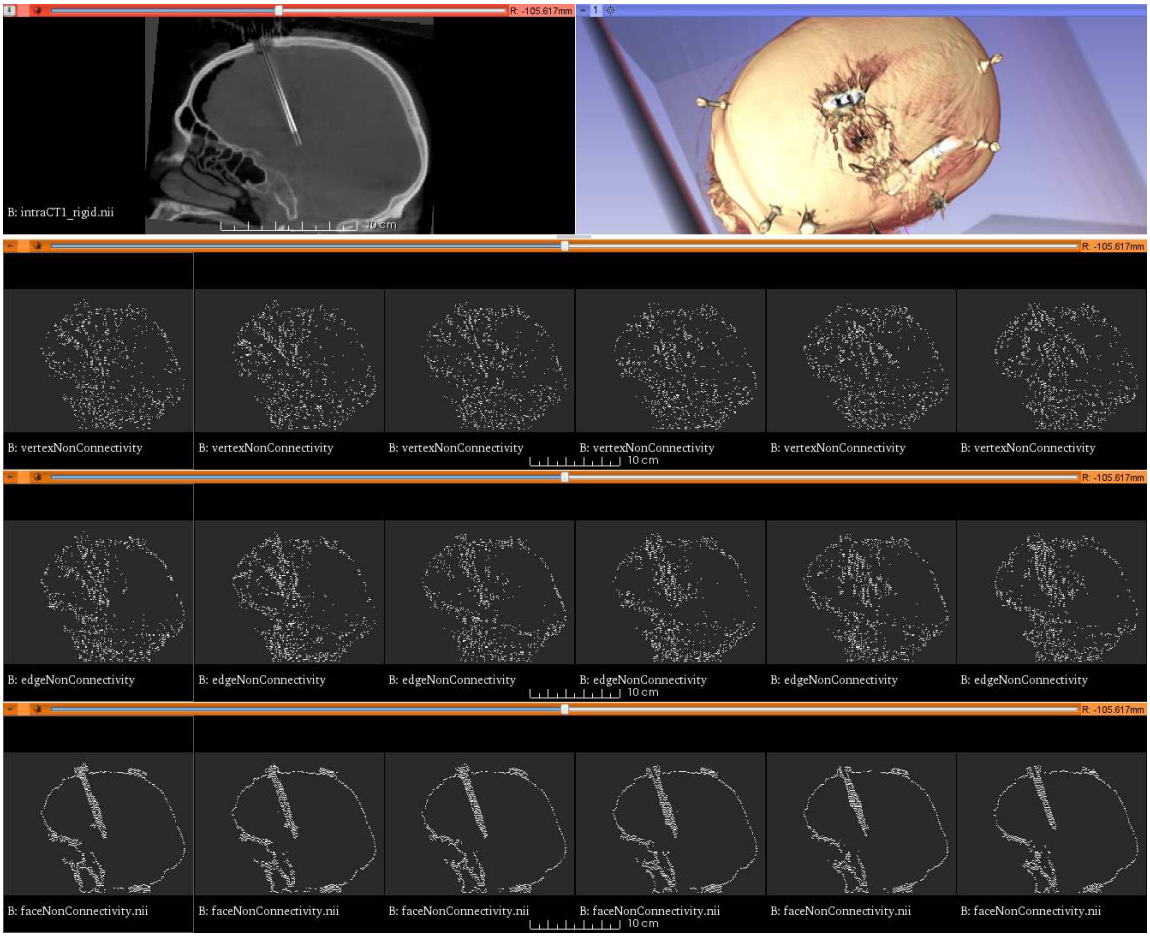


Figure 106: The distribution of the selected blocks in a brain CT scan, using different connectivity patterns. The results are depicted on six consecutive slices. From top to bottom row: sagittal CT slice (left) and volumetric rendering (right), selected blocks with "vertex" connectivity, selected blocks with "edge" connectivity, selected blocks with "face" connectivity. The "vertex" pattern results in a more uniform distribution, while the "face" pattern results in a higher block density near the lead and tissue boundaries.

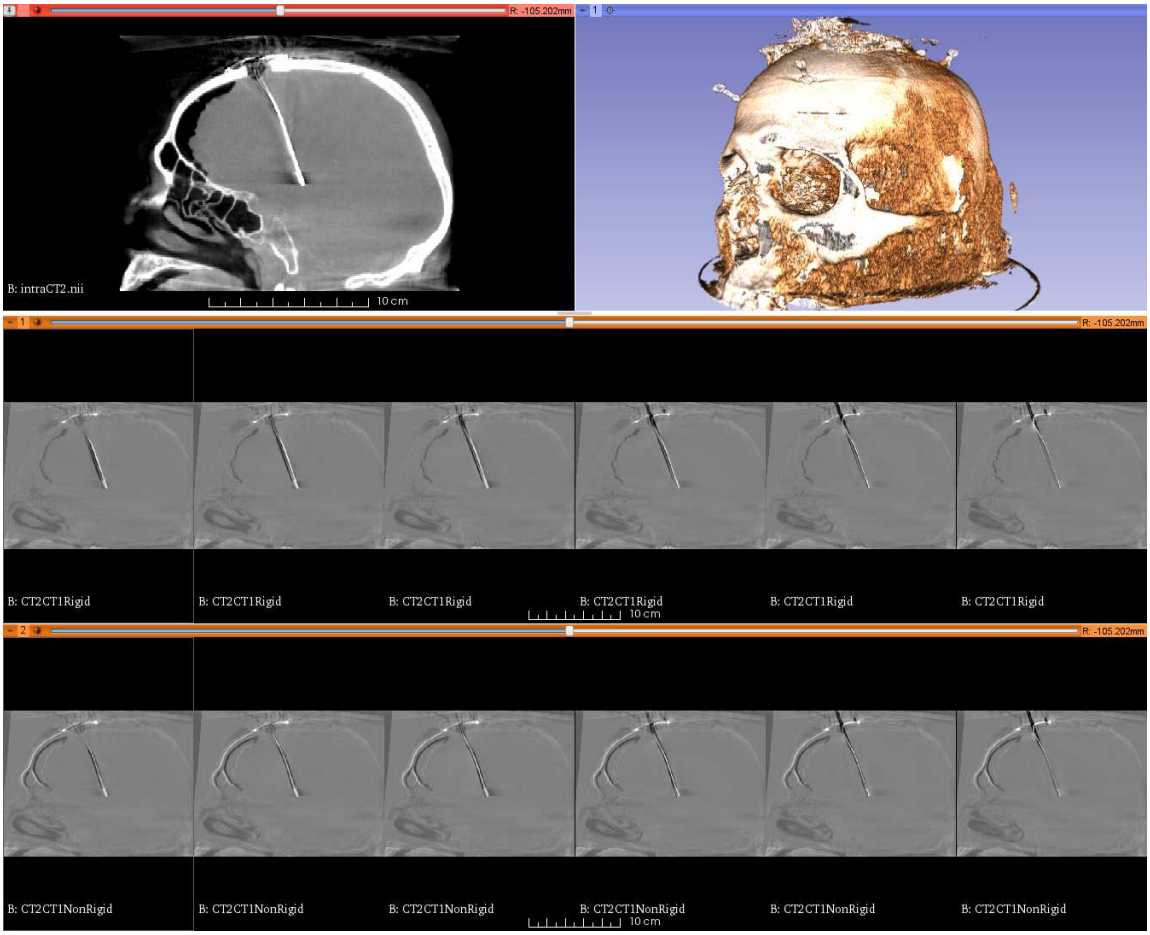


Figure 107: Qualitative results for case 1. Top row: intra-op O-arm CT2 (left) and its volume rendering (right). Middle row: Rigid registered O-arm CT1 subtracted from CT2. Bottom row: Deformable registered O-arm CT1 subtracted from CT2.

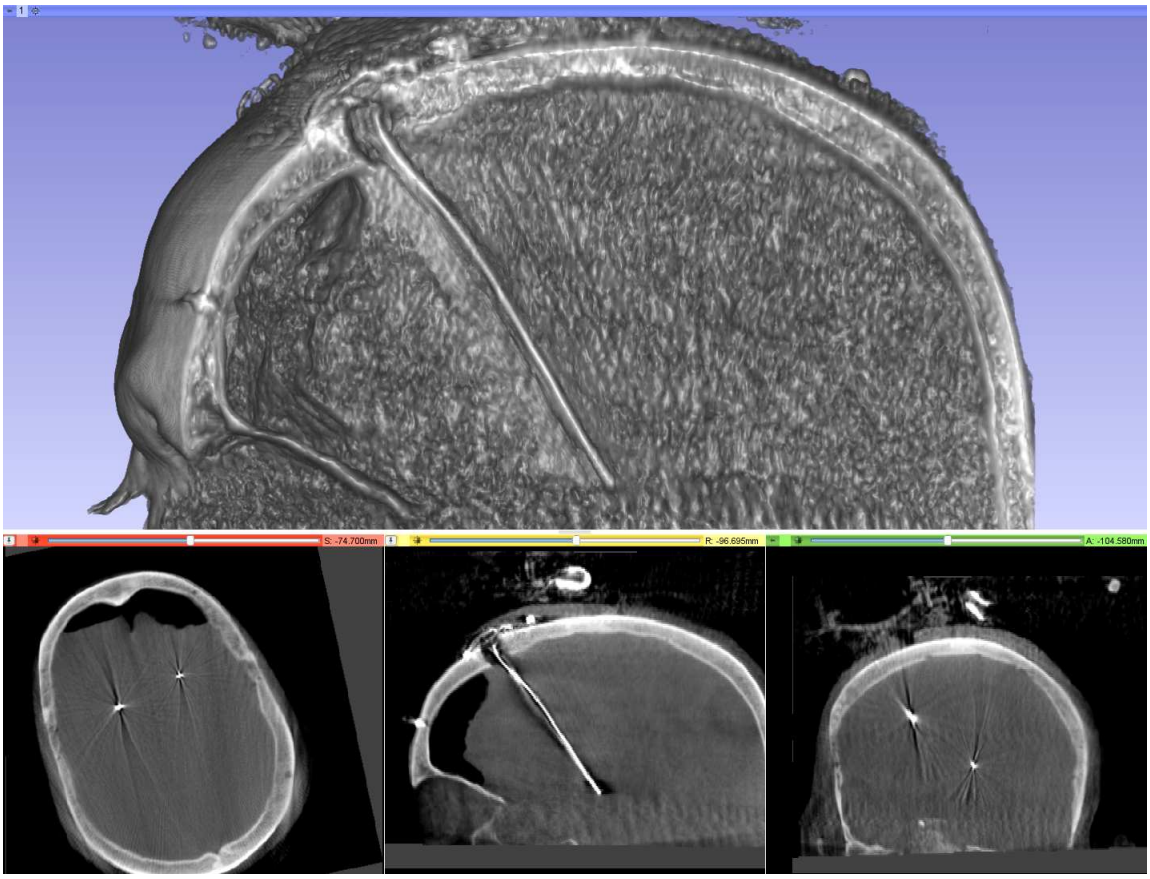
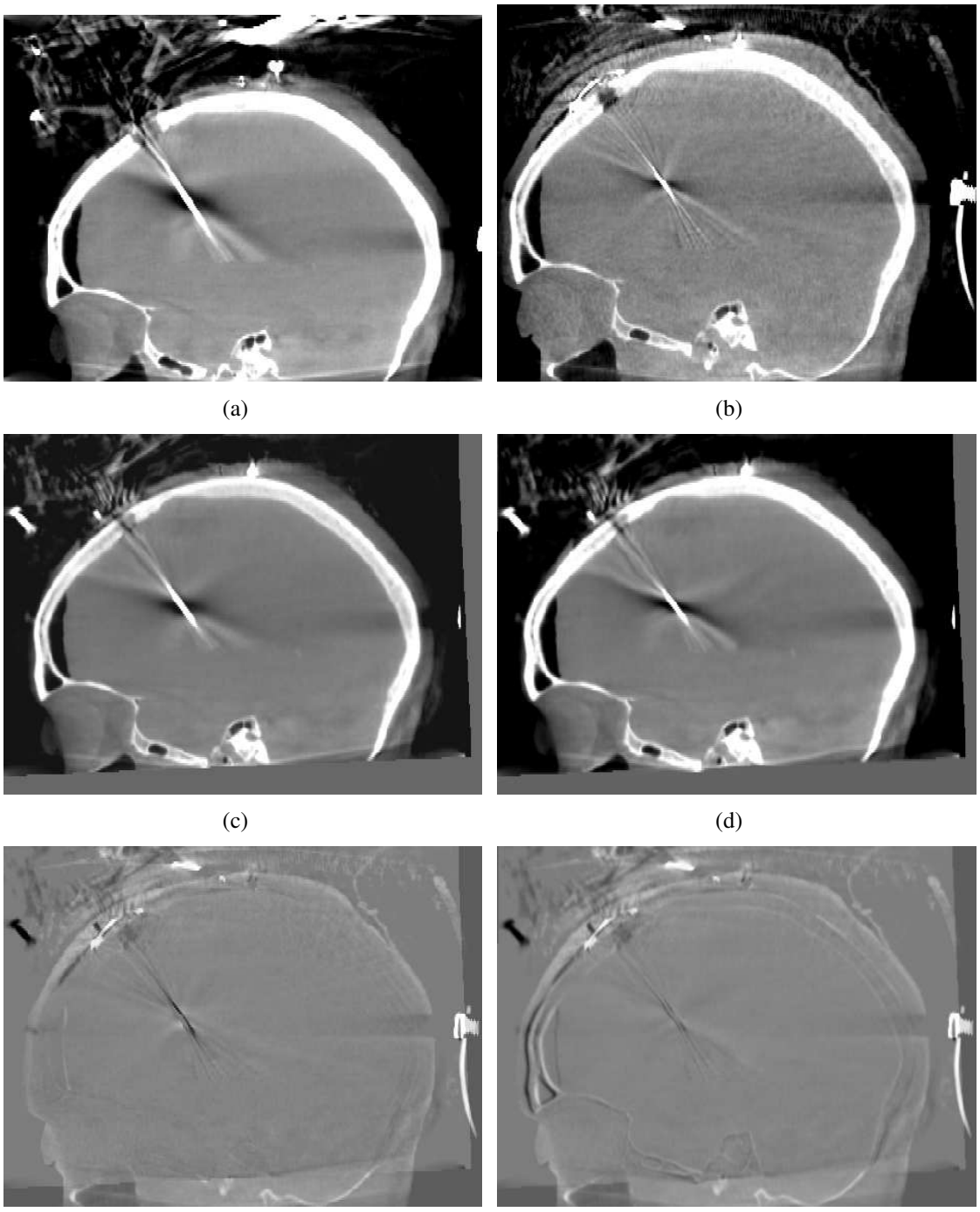


Figure 108: Deformable registered O-arm CT1 for patient 2. The shift is larger near the cortical structures and smaller in deep brain structures (tip of the lead). Top: cut section of a volume rendering. Bottom: axial, sagittal, and coronal slices.



(a) (b)

(c) (d)

(e) (f)

Figure 109: Qualitative evaluation results for case 3. (a)-(f) depict the same sagittal slice of the volumetric CT scan. (a): O-arm CT1; (b): O-arm CT2; (c): rigid registered O-arm CT1; (d): non-rigid registered O-arm CT1; (e): rigid registered O-arm CT1 subtracted from O-arm CT2; (f): non-rigid registered O-arm CT1 subtracted from O-arm CT2.

APPENDIX B

SOFTWARE USAGE

B.1 CBC3D USAGE

–image: multi-labeled segmented image.

–latticeSpacing: spacing of the BBC lattice (in mm).

- Range: [0.1, inf]

Default: 8

–fidelity: mesh fidelity.

- Range: [0.1, 1]

Default: 0.8

–multiple-fidelities: file that contains the fidelity values of each material.

–upsampleResolution: performs a global image up-sampling.

- 0: no up-sampling
- 1: two-fold up-sampling
- 2: four-fold up-sampling

Default: 0

–checkImageTopology: performs image topological checks and converts the non-manifold voxel connectivity into a manifold voxel connectivity.

- 0: no
- 1: yes

Default: 0

–checkMeshTopology: performs mesh topological checks and imposes additional refinement (local or global) to eliminate the non-manifold element connectivity. Applies only if checkImageTopology 1.

- 0: no

- 1: yes

Default: 0

–thresholdToRelabelDisconnectedImageRegions: threshold for relabeling the disconnected image regions.

- Range: (0, 0.01]

Default: 0.001

–thresholdToRelabelDisconnectedMeshRegions: threshold for relabeling the disconnected mesh regions.

- Range: (0, 0.1]

Default: 0.001

–hybridMesh: generates a mixed mesh from an adaptive tetrahedral mesh by merging clusters of tetrahedra into hexahedra and pyramids.

- 0: no
- 1: yes

Default: 0

–maxNumberSurfaceTrianglesOnPyramid: adjusts the topology on the mesh surface by allowing a pyramid to have 0, 1, 2, 3, or 4 surface triangles.

- Range: [0, 4]

Default: 0

–shapeFunctionType: element shape functions (for tetrahedra and hexahedra).

- 0: linear
- 1: quadratic

Default: 0

–outputSurfaceBCCMeshTriangles: prints the surface triangles of the BCC mesh.

–outputBCCMesh: prints the BCC mesh (.vtk).

–numIterations: number of smoothing iterations.

- Range: [0, $\inf$)

Default: 0

–flexibility: trade-off between the regularization energy defined by the stress energy of a linear biomechanical elastic model, and the similarity energy.

- Range: [0.1, 1]

Default: 1

–E: Young's modulus (N/mm^2).

- Range: $(0, \mathrm{inf})$

Default: 0.0021

–v: Poisson's ratio.

- Range: [0.1, 0.49]

Default: 0.45

–nonConnectivity: non-connectivity pattern for the extraction of the target points.

- 0: "vertex"
- 1: "edge"
- 2: "face"
- 3: "no"

Default: 0

–minDihedralAngle: minimum dihedral angle for smoothing.

- Range: $(0^\circ, 30^\circ]$

Default: 5°

–minScaledJacobian: minimum scaled Jacobian for smoothing.

- Range: [0, 1]

Default: 0.20

–scaleFactorForSearchRadius: scales the neighborhood to search for target points.

- Range: $(0, \mathrm{inf})$

Default: 1.5

–scaleFactorForEdges: scales the average length of the edges incident to a vertex.

- Range: (0, 5]

Default: 0.2

–scaleFactorForConfidence: scales the confidence of landmarks incident on low quality elements.

- Range: (0, 1]

Default: 0.2

–maxNumberScaledConfidenceIterations: maximum number of iterations with a scaled confidence.

- Range: [0, inf)

Default: 3

–numberLaplacianSmoothingIterations: number of iterations for Laplacian smoothing to improve mesh quality.

- Range: [0, inf)

Default: 0

–localTransformations: enables local transformations tom improve mesh quality.

- 0: no
- 1: yes

Default: 0

–outputSourcePoints: prints the source points used for smoothing (.vtk).

–outputTargetPoints1: prints the target points used for smoothing (.vtk).

–outputTargetPoints2: prints the target points used for smoothing in an image (e.g., .nrrd).

–outputSurfaceSmoothMeshTriangles: prints the surface triangles after smoothing (.vtk).

–outputSmoothMesh: prints the smoothened mesh (.vtk).

–numThreads: number of threads.

- Range: [1, inf)

Default: 1

–evaluateMeshQuality: prints the dihedral angles histogram and some quality statistics.

- 0: no
- 1: yes

Default: 0

–evaluateMeshVolume: computes the mesh volume, and checks if the sum of the volumes of the elements equals to the volume defined by the polyhedral domain.

- 0: no
- 1: yes

Default: 0

–evaluateMeshFidelity: computes the mesh fidelity using a HD metric.

- 0: no
- 1: yes

Default: 0

B.2 A-PBNRR USAGE

–movingImage: moving (preoperative) image.

–fixedImage: fixed (intraoperative) image.

–segmentedImage: segmentation of the moving image.

–halfBlockSizeX: size of half block in X image direction (in voxels).

- Range: [1, inf)

Default: 1

–halfBlockSizeY: size of half block in Y image direction (in voxels).

- Range: [1, inf)

Default: 1

–halfBlockSizeZ: size of half block in Z image direction (in voxels).

- Range: [1, inf)

Default: 1

–halfWindowSizeX: size of half window in X image direction (in voxels).

- Range: [1, inf)

Default: 5

–halfWindowSizeY: size of half window in Y image direction (in voxels).

- Range: [1, inf)

Default: 5

–halfWindowSizeZ: size of half window in Z image direction (in voxels).

- Range: [1, inf)

Default: 5

–simMetric: similarity metric for block matching.

- 0: Normalized Cross Correlation (NCC)
- 1: Normalized Mutual Information (NMI)

Default: 0

–nonConnectivity: non-connectivity voxel pattern for feature selection.

- 0: Vertex non-connectivity
- 1: Edge non-connectivity
- 2: Face non-connectivity

Default: 0

–meshMethod: mesh generation method.

- 0: CBC3D
- 1: Delaunay
- 2: LD
- 3: Structured Hex

Default: 0

–selectionFraction: fraction of selected image blocks from total number of image blocks.

- Range: (0, 1]

Default: 0.05

–rejectionFraction: fraction of rejected image blocks from number of selected image blocks.

- Range: (0, 1]

Default: 0.25

–tradeOff: controls the trade-off between regularization energy defined by the stress energy of a linear elastic model and the similarity energy.

- Range: (0, 1]

Default: 1

–numApproxSteps: number of approximation (outlier rejection) iterations.

- Range: [1, inf)

Default: 5

–numInterSteps: number of interpolation iterations.

- Range: [0, inf)

Default: 5

–numAdaptiveIterations: number of adaptive iterations.

- Range: [1, inf)

Default: 1

–zeroBlocksFraction: minimum ratio of blocks without correspondence to selected image blocks.

- Range: [0, 1]

Default: 0.01

–shapeFunctionType: type of shape function for interpolation.

- 0: Linear
- 1: Linear w/ ESF
- 2: Quadratic
- 3: Quadratic w/ ESF

Default: 0

–tumorResection: tumor resection flag.

- 0: off
- 1: on

Default: 0

–HDEvaluation: evaluation of registration accuracy with a Hausdorff Distance metric.

- 0: off
- 1: on

Default: 0

–outputDeformationField: output deformation field; the field maps points in the fixed image to homologous points in the moving image.

–outputImage: output registered image.

–numThreads: number of threads.

- Range: [1, inf)

Default: 1

–E1: Young modulus (N/mm^2) for material with label 1 (brain parenchyma).

- Range: (0, inf)

Default: $2.1e\text{-}4$

–v1: Poisson ratio for material with label 1 (brain parenchyma).

- Range: (0, $\inf$)

Default: 0.45

–E2: Young modulus (N/mm^2) for material with label 2 (tumor).

- Range: (0, $\inf$)

Default: $2.1e\text{-}3$

–v2: Poisson ratio for material with label 2 (tumor).

- Range: (0, $\inf$)

Default: 0.45

–CBC3DSpacing: lattice spacing (in mm) for CBC3D mesh generation.

- Range: (0, $\inf$)

Default: 8

–CBC3DFidelity: mesh fidelity for CBC3D mesh generation.

- Range: [0.1, 1]

Default: 0.85

–CBC3DNumIterations: smoothing iterations for CBC3D mesh generation.

- Range: [0, $\inf$)

Default: 0

–CBC3DMultipleFidelities: multi-fidelities for CBC3D mesh generation.

–DelaunayDelta: element size for Delaunay mesh generation.

- Range: (0, $\inf$)

Default: 5

B.3 CDT3D USAGE

–plc: input piecewise linear complex (stl or vtk).

–sort: sorting flag for input boundary points.

- 0: no sorting
- 1: random permutation
- 2: sorting based on BRIO and Hilbert curve

Default: 2

–brec: boundary recovery flag.

- 0: do not recover boundary
- 1: recover only boundary partitions
- 2: full boundary recovery

Default: 2

–ref: refinement flag.

- 0: do not refine
- 1: refine

Default: 1

–cdf: distribution function multiplier. The distribution function specifies the desired element size. The distribution function originally determined from the average of the surrounding boundary edge lengths is multiplied by cdf.

- Range: [0.5, 3]

Default: 1.2

–cdfn: nearby node factor. Nodes are considered too close if the distance between them is less than their average node distribution functions multiplied by cdfn.

- Range: [0.5, 2]

Default: 0.7

–cdfm: distribution function weighting factor. The node distribution function for new nodes is averaged with the minimum nearby node distribution functions. This factor is the weighting for the minimum contribution. Increasing cdfm above 0.0 will generally reduce the growth of element size from small to larger elements and increase the total number of grid nodes generated. It will have little or no effect if all the boundary edges are nearly the same size.

- Range: [0, 1]

Default: 0

–cdff: satisfied edge length multiplier. An edge is considered satisfied if its length divided by this factor is less than the average of the node distribution function at the edge end-nodes.

- Range: [1.1, 3]

Default: 1.5

–cdfr: maximum geometric growth rate. Used as the advancing-front growth limit. The element size for new nodes is limited to be less than the physical size of the local front advanced from multiplied by cdfr.

- Range: [1, 3]

Default: 1.1

–dfmax: maximum distribution function. The distribution function specifies the point spacing in the field. If dfmax < 0 then do not limit the maximum distribution function. If dfmax = 0 then limit the maximum distribution function to the maximum value determined from the boundary surface grid. If dfmax > 0 then limit the maximum distribution function to dfmax.

- Range: [-1, 1e+19]

Default: -1

–angmax: satisfied dihedral element angle. An element is considered satisfied if it has a maximum angle less than angmax and if all of its edge lengths are satisfied.

- Range: [120, 179.9]

Default: 150

–angqmax: low quality dihedral element angle. Quality improvement is repeated if there are elements with a maximum angle greater than angqmax. Also an element will not be created inside a boundary element if it would create a maximum element angle greater than angqmax.

- Range: [90, 179.9]

Default: 160

–angqmsk: masking dihedral element angle. Special quality improvement operations are performed on elements with a maximum angle greater than angqmsk.

- Range: [90, 179.9]

Default: 120

–angqual: final quality dihedral element angle. Local reconnection for quality improvement is performed on elements with a maximum angle greater than angqual.

- Range: [70, 179.9].

Default: 120

–angdbs: boundary sliver dihedral element angle. All boundary elements with a dihedral angle greater than angdbs are considered slivers and therefore deleted. The boundary surface triangles are reconnected to delete boundary slivers.

- Range: [120, 179.9]

Default: 160

–angdfs: quality field sliver dihedral angle. During quality improvement, all field elements with an angle greater than angdfs are considered sliver elements and are minimized by inserting a node near the element centroid. Only applicable if the quality field sliver deletion flag is on (mdfs = 1).

- Range: [120, 179.9]

Default: 165

–angrbfdd: discontinuous dihedral surface angle. Dihedral angle between two adjacent faces used to limit boundary surface recovery reconnection. The boundary surface triangulation will not be reconnected if the result is a dihedral angle between the two faces which is less than angrbfdd.

- Range: [120, 179.9]

Default: 150

–angrbfmxp: maximum planar face angle. Planar face angle used to limit boundary surface recovery reconnection. The boundary surface triangulation will not be reconnected if the result is a greater maximum angle and if the reconnected maximum angle is greater than angrbfmxp.

- Range: [60, 179.9]

Default: 140

–angqbfm: maximum low-quality planar surface angle. Planar surface angle for boundary triangular faces used to check for a low-quality boundary surface triangulation. If the planar angle for a vertex of any face is greater than angqbfm, then the triangulation is considered low-quality and mrec4 is set to a value of 1.

- Range: [0, 180]

Default: 160

–mrecm: local-reconnection flag.

- 1: use Delaunay criterion
- 2: use combined Delaunay and min-max Laplacian edge weight
- 3: use combined Delaunay and min-max skew. (not yet implemented)

Default: 2

–mrecm2: local-reconnection flag for traversing the element link-list (applies only in reconnection with 2-3, and 3-2 flips).

- 0: reconnect the grid in a single traversal
- 1: reconnect the grid in repetitive traversals

Default: 0

–mrecqm: quality local-reconnection flag. Applies to all quality improvement except the last pass.

- 2: use a combined Delaunay and min-max Laplacian edge weight
- 3: use a combined Delaunay and min-max skew. This option typically results in a slightly improved grid quality at the expense of an increase in required CPU time

Default: 2

–mrecqmf: final quality local-reconnection flag. Applies only to the last pass of quality improvement.

- 2: use a combined Delaunay and min-max Laplacian edge weight
- 3: use a combined Delaunay and min-max skew (not yet implemented).

Default: 2

–mrec4: 4-4 element local-reconnection flag.

- 0: reconnect 4-4 element only during final quality improvement
- 1: reconnect 4-4 element combinations during grid generation

Default: 0

–mrec4q: quality 4-4 element local-reconnection flag (applies when mrec4 = 1).

- 0: reconnect 4-4 element with a lower final quality
- 1: reconnect 4-4 element with higher final quality

Default: 1

–pivot: face normal flag.

- 0: compute face normal without sorting its edges (faster)
- 1: compute a face normal from its sortest edges (more accurate)

Default: 1

–ngen: maximum number of grid passes.

- Range: [0, 10000000].

Default: 10000

–mdse: small edge deletion flag.

- 0: do not delete small edges
- 1: delete small edges

Default: 1

–cdse: small edge factor. Small edges are deleted if the small edge deletion flag is on (mdse = 1).

- Range: [0, 0.7]

Default: 0.25

–outmesh: output mesh (vtk).

–maxStack: max soft stack size (MB).

–loc: point location flag.

- 0: use exact arithmetics to locate points
- 1: use non-exact arithmetics for points located close to a face or an edge and insert them as they were exact on a face or an edge

Default 0

–loctol: tolerance for loc = 1.

- Range: [0, 0.001]

Default: 1e-5

–qstat: grid quality statistics flag.

- 0: suppress
- 1: print

Default: 0

–nqual: number of quality improvement passes. If nqual = 0 then all quality improvement is skipped.

- Range: [0,10]

Default: 2

–nqualf: number of quality improvement repeats. Maximum number of times to repeat quality improvement if grid quality is too low.

- Range: [0,10]

Default: 1

–nsmth: number of smoothing iterations.

- Range: [0, 10]

Default: 3

–csmth: smoothing coefficient.

- Range: [0, 1]

Default: 0.5.

–vsmthb: smoothing coefficient reduction factor for all nodes adjacent to a boundary.

- Range: [0, 1]

Default: 0.5

–msmth: smoothing flag.

- 1: use optimal placement smoothing initially

- 2: use centroid averaging smoothing initially

Note: If the selected type produces an invalid grid then the other type is attempted. *Default: 1*

–mdbs: boundary sliver deletion flag.

- 0: do not delete boundary slivers
- 1: delete boundary slivers and reconnect boundary. The boundary surface triangles are reconnected to delete boundary slivers.

Default: 0

–mdfs: quality field sliver deletion flag.

- 0: do not add nodes to delete field sliver elements during quality improvement
- 1: add nodes to delete field sliver elements during quality improvement

Default: 1

–ndbs: maximum number of repeats for deleting slivers from boundary (applies if mdbs = 1).

- Range: [0, 100]

Default: 5

–ndfs: maximum number of repeats for adding field nodes to delete slivers (applies if mdfs = 1).

- Range: [0, 100]

Default: 10

–nthreads: number of threads for grid refinement. *Default: 1*

–nthreadsq: number of threads for grid quality improvement. *Default: nthreads*

–nbuckets: number of buckets for data-decomposition in grid refinement.

- Range: [1, nelements]

Default: nthreads

–nbucketsq: number of buckets for data-decomposition in grid quality improvement.

- Range: [1, nelements]

Default: nthreadsq

–frbtransf: fraction of buckets to transfer among threads for load balancing in grid refinement.

- Range: [0,1]

Default: 0.3

–cbtransf: coefficient to reduce the number of buckets transfered between a pair of threads in grid refinement.

- Range: (0,1]

Default: 1

–frbtransfq: fraction of buckets to transfer among threads for load balancing in grid quality improvement.

- Range: [0,1]

Default: 0.3

–cbtransfq: coefficient to reduce the number of buckets transfered between a pair of threads in grid quality improvement.

- Range: (0,1]

Default: 1

–ordt: orders the elemnts in the link-list into two groups (first the active elements, and then the inactive elements) to improving the performance. Applies in grid refinement.

- 0: do not order
- 1: order

Default: 1

–ordtq: orders the elemnts in the link-list into two groups (first the active elements, and then the inactive elements) to improving the performance. Applies in grid quality improvement.

- 0: do not order
- 1: order

Default: ordt

–sortp: sort elements with BRIO and Hilbert curve before data-decomposition in grid refinement.

- 0: off
- 1: on

Default: 0

–sortpq: sort elements with BRIO and Hilbert curve before data-decomposition in grid quality improvement.

- 0: off
- 1: on

Default: 0

–blockmul: multiplies the sizeof(object) to pre-allocate a continuous block in memory.

- Range: [1, 10e9]

Default: 10e5

–pint: pins a thread to a specific CPU.

- 0: off
- 1: on

Default: 1

–nminpts: minimum number of generated candidate points to achieve convergence in grid refinemenent.

- Range: [0,1000]

Default: 1

–sliv: counts the slivers after the completion of the grid refinement.

- 0: off
- 1: on

Default: 0

VITA

Fotios Drakopoulos
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

Fotios Drakopoulos is working in Real-Time aspects of Engineering Geometry for applications related to Aerospace Engineering and Medical Image Computing. He has 12 years of experience in software development (including commercial, open-source, and academic software) in Europe and United States, and his work spans a spectrum of applications, from health care to civil and aerospace engineering. He is the first creator of CDT3D, a parallel unstructured grid generator to implement exascale-era meshing for NASA's CFD 2030 vision. In health care, namely anatomic modeling, he developed Image-to-Mesh (I2M) conversion techniques for Brain Tumor Resection and Deep Brain Stimulation to help compute brain deformation. Additionally, he fine-tuned his I2M conversion approach to assist in applying Numerical Simulation of Cerebral Aneurysm by Flow Diversion and Blood Flow simulation for Arteriovenous Malformations. His I2M tools also helped him to expand his research in fusion of brain images, and his recent work has measurable impact in improving the accuracy of Non-Rigid Registration for tumor resection in Image-Guided Neurosurgery. In the past, he worked as a software engineer in Computer Control Systems Engineering Research and Design S.A, where he developed commercial FEM software for the modeling, analysis, and dimensioning of multi-story composite buildings. He has published 16 peer-reviewed papers, twice received the award for best paper MSVE Capstone Conference in Medical simulations (2014, 2016), twice received the outstanding research assistant award in Computer Science from Old Dominion University, and accepted a 3-year (maximum allowance) Modeling and Simulation fellowship to support his research.

Typeset using LaTeX.