THE UNIVERSITY OF
SYDNEY

Locally Optimal Delaunay-refinement and Optimisation-based Mesh Generation

DARREN ENGWIRDA

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Mathematics and Statistics
Faculty of Science
The University of Sydney

September 2014

Abstract

The field of mesh generation concerns the development of efficient algorithmic techniques to construct high-quality tessellations of complex geometrical objects. In this thesis, I investigate the problem of unstructured simplicial mesh generation for problems in $\mathbb{R}^2$ and $\mathbb{R}^3$, in which meshes consist of collections of triangular and tetrahedral elements. I focus on the development of efficient algorithms and computer programs to produce high-quality meshes for planar, surface and volumetric objects of arbitrary complexity. I develop and implement a number of new algorithms for mesh construction based on the Frontal-Delaunay paradigm – a hybridisation of conventional Delaunay-refinement and advancing-front techniques. I show that the proposed algorithms are a significant improvement on existing approaches, typically outperforming the Delaunay-refinement technique in terms of element shape- and size-quality, while offering significantly improved theoretical robustness compared to advancing-front techniques. I verify experimentally that the proposed methods achieve the same theoretical element shape- and size-bounds typically associated with conventional Delaunay-refinement techniques. An implementation of the new Frontal-Delaunay algorithms, in addition to conventional Delaunay-refinement techniques, are provided in the new mesh generation package `JIGSAW`. In addition to mesh construction, methods for mesh improvement are also investigated. I develop and implement a family of techniques to improve the element shape quality of existing simplicial meshes, using a combination of optimisation-based vertex smoothing, local topological transformation and vertex insertion. These operations are interleaved according to a new priority-based schedule, and I show that the resulting algorithms are competitive with existing state-of-the-art approaches in terms of mesh quality, while offering significant improvements in computational efficiency. An implementation of the suite of new mesh optimisation algorithms for the improvement of planar, surface and volumetric meshes are provided in the `JITTERBUG` package.

Acknowledgements

This research was carried out in the Department of Mathematics and Statistics at the University of Sydney with the support of an Australian Postgraduate Award. I am grateful to many within the department for the illuminating discussions, suggestions and contributions made throughout. Foremost, I would like to thank my supervisor, David Ivers, who has not only imbued the project with a wealth of knowledge and rigour, but also gave me the academic freedom and encouragement necessary to pursue such work from the outset. To the many friends and family, who have both cheerfully supported and ultimately tolerated my studies – I couldn't have done it without you. And finally to Sara, who ceaselessly sustains and inspires – I fear you are owed a debt immeasurable.

Contents

Chapter 1

Introduction

Computational modelling and simulation is a critical aspect of modern scientific research and industrial design, underpinning a diverse range of applications – from computer graphics and animation to complex numerical simulations of physical phenomena, such as fluid dynamics and structural mechanics. These computational techniques are based on a discrete representation of the underlying geometry, in which the objects and domains of interest are tessellated into *meshes* of simple elements. The study of the properties, construction and maintenance of these tessellated models is spanned by the field of *mesh-generation*.

Modern mesh generation is concerned with the development of efficient and automatic algorithms to construct and maintain high-quality meshes for complex objects and domains. Meshes are expected to conform to a number of often divergent requirements – adherence to complex geometrical features, support for high spatial resolution in areas of interest and maximum sparsity elsewhere, facilitation of adaptive and incremental refinement and, significantly, preservation of *optimal* element geometry. The development of efficient techniques and algorithms that offer provable guarantees on the worst-case element quality and optimality represents a major challenge in contemporary mesh generation research.

This dissertation concerns the problem of unstructured simplicial mesh generation for two- and three-dimensional problems, in which meshes consist of collections of triangular and tetrahedral elements embedded in Euclidean space. I focus on the development of efficient algorithms and computer programs to produce high-quality meshes for *planar*, *surface* and *volumetric* objects of arbitrary complexity, suitable for subsequent modelling and numerical simulation. In this work, I concentrate on two key areas: (i) the development of fundamental meshing algorithms based on the Delaunay tessellation, and (ii) the development of techniques for mesh improvement, utilising local geometrical and topological optimisation operations.

In the remainder of this chapter I offer a brief introduction to the notion of meshes, and discuss the manner in which the optimality of these structures can be assessed and measured. Additionally, I briefly review the development of a range of methods for mesh generation, contrasting the various advantages and disadvantages associated with

existing techniques. Finally, I outline the structure and objectives of this dissertation, summarising the major results and contributions to be presented in subsequent chapters.

1.1 Meshes

Meshes are collections of simple geometrical elements that tessellate a set of vertices in space. In this work, I am interested only in geometrical problems, in which the vertices lie in two- or three-dimensional Euclidean space, such that $X \subset \mathbb{R}^2$ or $X \subset \mathbb{R}^3$. Meshes can be broadly categorised according to their underlying character, examining the structure of their topology, the choice of underlying element type and the nature of their inter-element adjacencies.

Meshes are classed as being of either the *structured* or *unstructured* variety, referring to the regularity of their underlying topology. Structured meshes consist of highly regular configurations, in which the adjacency of all vertices and elements is represented in terms of a uniform template. Due to their simplicity, structured meshes support *implicit* adjacency queries, where connectivity is calculated using simple arithmetic operations. Conventional 'grids', based on a uniform subdivision of Cartesian space, are illustrative of common structured mesh types. Unstructured meshes, in contrast, do not incorporate regular vertex or element connectivities, requiring the *explicit* storage of vertex and element adjacency information for each entity in the mesh. Such meshes are typified by triangle-based structures, in which a domain is decomposed into a collection of irregular triangle or tetrahedral elements, although generalised unstructured meshes can also be formulated in terms of arbitrary polygonal or polyhedral element types.

Due to the regularity of their indexing and connectivity, structured meshes impose low computational storage requirements and can facilitate the development of highly efficient numerical methods. On the downside, their stringent requirements on topological regularity can cause serious problems when dealing with complex geometrical constraints or variable spatial resolution, typically restricting the use of such meshes to problems specified on simple geometrical domains. Unstructured meshes, on the other hand, offer far greater geometrical flexibility, and are generally preferred when tessellating the complex objects and domains that arise when modelling problems deriving from industrial or naturally occurring processes. Despite the additional computational cost that unstructured formulations may impose on a *per-element* basis, the flexibility of variable and adaptive spatial resolution can often lead to significant overall computational savings, due to a reduction in total element count.

Meshes are also classified according to their inter-element connectivities, with so-called *conforming* meshes requiring that adjacent elements intersect along a common edge or face. Conversely, *non-conforming* mesh types support adjacency amongst multi-element 'patches', in which neighbouring elements share only partial edge- or face-based connectivity. While non-conforming strategies offer clear advantages in terms of geometrical and topological flexibility, the resulting fractional element connectivities dictate that only sophisticated and potentially computationally expensive numerical methods can be supported. In this study, in the interests of general applicability, I pursue the

development of conforming unstructured meshing techniques only.

Unstructured meshes can be built using a variety of polygonal and polyhedral element types, including *triangles*, *quadrilaterals*, *tetrahedrons*, *pyramids*, *wedges* and *hexahedrons*. Meshes are often homogeneous, consisting of a single element type, although it's possible to consider *mixed-element* meshes incorporating various combinations of the basic types. In this work, I restrict my attention to so-called *simplicial* meshing techniques, in which the basic element of a mesh in $\mathbb{R}^d$ is the *d-simplex* – the simplest convex full-dimensional polyhedron that can be constructed from a minimal set of points in $\mathbb{R}^d$. In two- and three-dimensions these elements are simply the well known triangle and tetrahedron, respectively. Simplicial tessellations exhibit a number of convenient theoretical properties that facilitate the development of high-quality meshing techniques. A formal discussion of these properties is deferred until Chapter 2. The development of theorems and techniques supporting provable guarantees for general non-simplicial mesh generation remains a significant open problem in computational geometry and is not pursued further in this study.

Beyond definitions of the basic geometrical and topological characteristics of meshes, it is necessary to discuss the mechanisms through which the *quality* and *optimality* of an unstructured mesh can be measured. The problem of constructing optimal meshes and tessellations has been studied by a number of authors, including, for example, Cheng, Dey and Shewchuk [13, 52], Freitag, Ollivier-Gooch and Knupp [23, 24], and Alliez, Cohen-Steiner, Yvinec, and Desbrun [1]. In general, it is noted that a *high-quality* mesh satisfies demands on both the size and shape of the underlying elements. In [13, Chapter 1] Cheng, Dey and Shewchuk provide a broad summary of the effects of element shape and size, making the following observations:

- Elements with large dihedral angles introduce correspondingly large errors in numerical approximations of differential operators. It is noted that as the worst-case element-wise dihedral angles approaches 180° the errors in the gradients computed using piece-wise linear interpolants becomes unbounded. See Synge [53] and Babuška and Aziz [2] for a discussion of this effect for triangular elements and Křížek [31] for tetrahedral elements.
- Elements with small dihedral angles lead to poorly conditioned numerical integration schemes, in turn compromising the conditioning of the large linear systems that arise in many finite-volume and finite-element methods. This behaviour has been explored extensively within the numerical methods community – see, for example, studies by Fried [26].
- Highly skewed elements enclosing relatively small volumes can place severe restrictions on the maximum allowable time-step when solving time-dependent problems using explicit integration techniques. The nature of the restrictions are a function of the characteristics of the underlying differential equations and are particularly troublesome for some non-linear hyperbolic systems, such as the Navier-Stokes equations of fluid dynamics – see the Courant-Friedrichs-Lewy constraints [15] discussed by, for example, Lax and Wendroff [32] and LeVeque [33].
- Adaptation of local mesh resolution based on solution behaviour is desirable for optimal

results, providing high resolution and accuracy in areas of interest, while minimising overall computational costs through the adoption of sparse representations in smooth regions.

These considerations show that *high-quality* meshes are required to strike a balance between a number of competing criteria – supporting strong variability in spatial resolution while also preserving a range of constraints on element shape.

1.2 Methods for Mesh Generation

Interest in mesh generation coincided with early developments in numerical modelling and simulation in the 1970's and often focused on the application of the finite-element method to problems in structural mechanics. The field of mesh generation has evolved rapidly over the intervening decades, seeing the development of a number of important techniques and algorithms. The vast majority of modern meshing techniques fall into one of three basic categories: *advancing-front* methods, *grid-overlay* techniques and *Delaunay-based* approaches. Significant research has also focused on the development of *mesh-improvement* techniques, designed to enhance the quality of existing tessellations.

1.2.1 Grid-overlay Techniques

Grid-overlay techniques are arguably the simplest form of mesh generation, being based on an underlying structured mesh. While a number of variations exist in the literature, a typical approach seeks to *trim* and *warp* the elements of a structured 'overlay' grid to conform to the geometry of the domain to be meshed. While such methods can be based on simple uniform Cartesian meshes, superior techniques supporting spatial adaptivity can be realised through the use of non-uniform overlay meshes. Commonly, a semi-structured mesh, such as a *quadtree* or *octree* is used to construct the overlay mesh – recursively decomposing the axis-aligned bounding-box of the domain into a series of rectangular elements. These tree-based structures will be discussed in further detail in Chapter 2. See Figure 1.1 for an illustration of a quadtree-based overlay meshing algorithm in action.

The simplicity, and corresponding efficiency of grid-overlay meshing techniques are the main advantages of these methods, which are beset by a number of issues stemming from the potential incompatibilities and misalignment between the underlying structured grid and the geometry of the domain to be meshed. Typically, elements of sub-optimal shape are introduced near geometrical constraints, depending on alignment with the overlay mesh. Further alignment issues also often arise when attempting to refine these meshes along solution contours or other internal features. These techniques are also reported to often result in significant *over-refinement*, producing meshes that include too many elements. Grid-overlay techniques, based on quadtree or octree overlays, have been used in a number of meshing studies, including the early work of Yerry and Shephard [55, 56]. These methods also led to the development of the first *provably-good* meshing algorithm for general polygonal domains, due to Bern, Eppstein and Gilbert [3], in which

guarantees on the worst-case element angles were achieved, such that $18.4^\circ \leq \theta(\tau) \leq 153.2^\circ$, where $\theta(\tau)$ is the distribution of element-wise plane-angles in a triangulation $\mathcal{T}$. Mitchell and Vavasis also used tree-based overlay methods to construct size-optimal meshing algorithms for the three-dimensional [37] and d-dimensional cases [38].

1.2.2 Advancing-front Strategies

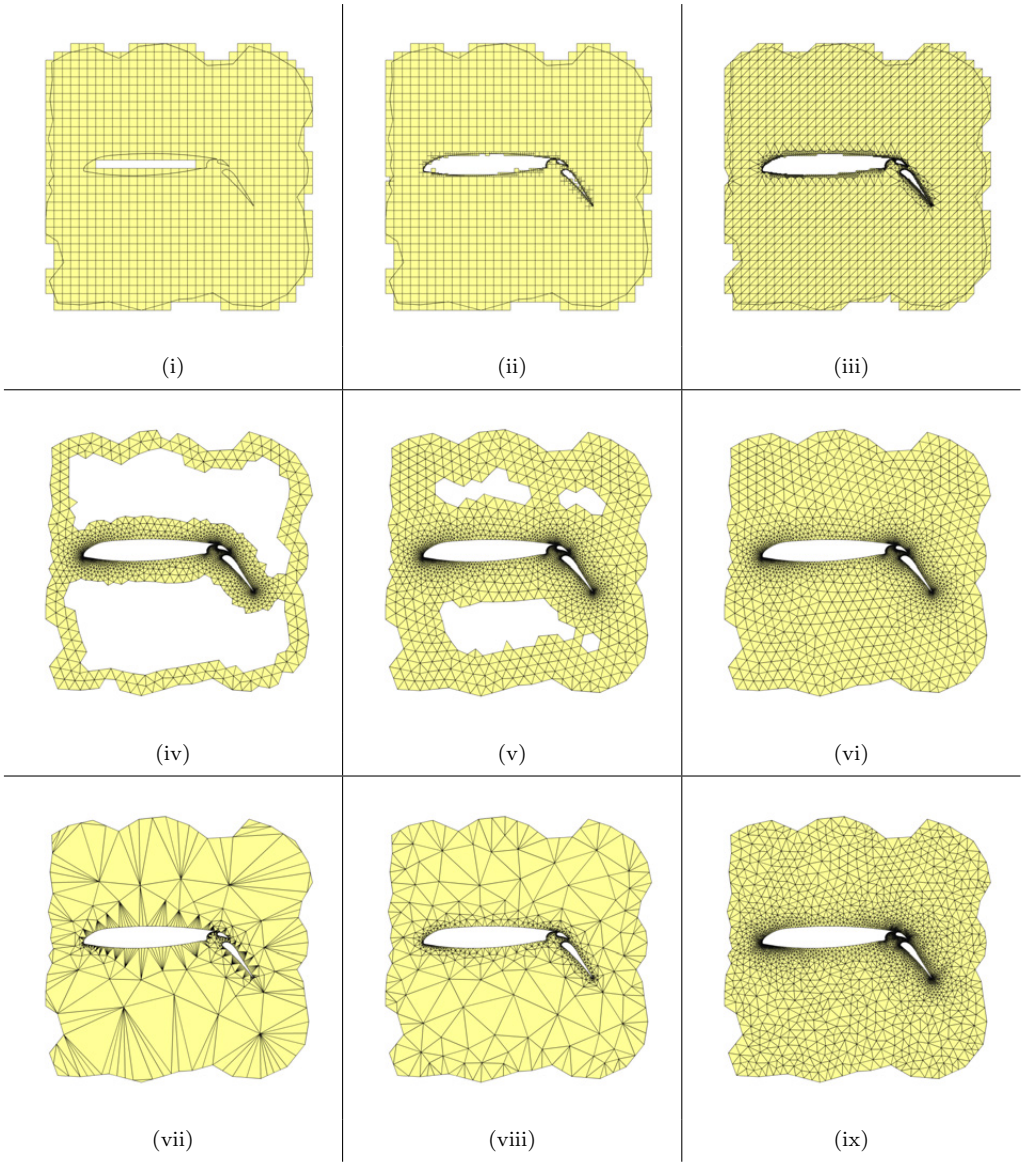
Advancing-front meshing strategies are based on an incremental paradigm, in which elements are created one-by-one, starting from a a set of *frontal* facets, initialised on the boundary of the domain to be meshed. The mesh is *marched* inwards layer-by-layer, with new elements created by carefully positioning vertices adjacent to the facets in the frontal set. Following the placement of new elements, the set of frontal facets is updated to incorporate the updated mesh topology. Elements are added incrementally in this fashion until a complete tessellation of the domain is obtained. For two-dimensional problems, advancing-front methods are known to often produce meshes with very high shape quality in practice, especially near the boundaries of the domain, where highly structured layer-wise configurations are typically achieved. The worst elements are typically generated where multiple fronts coalesce. See Figure 1.1 for an illustration of the frontal meshing procedure.

Despite excellent practical performance, conventional advancing-front methods are heuristic in nature and, as a result, have so-far resisted attempts to demonstrate meaningful theoretical guarantees. This lack of theoretical robustness becomes especially obvious when dealing with problems in higher dimensions, where, due to a number of topological constraints that will be outlined in Chapter 2, it is not necessarily possible to ensure that a valid tessellation can be constructed at each iteration of the frontal procedure for all inputs. Advancing-front methods are especially popular in the Computational Fluid Dynamics (CFD) community, where the preservation of element quality adjacent to boundary surfaces is a core consideration. An early frontal method was introduced by Alan George in his dissertation [27] leading to the development of a number of high-quality algorithms, such as those introduced by Peraire, Peiró and Morgan [40] and Löhner and Parikh [34]. A number of authors, including Schöberl [48] and Rypl [45, 46], have generalised the advancing-front method to tackle surface and volumetric meshing problems in three-dimensions. Schreiner, Scheidegger, Fleishman and Silva [49] have developed frontal methods for surface meshing applications in computer graphics. Blacker and Stephenson [4] and Rypl [47], amongst others, have also shown that advancing-front methods can be used to construct high-quality quadrilateral and mixed-element triangular-quadrilateral meshes for planar and surface geometries.

1.2.3 Delaunay-based Methods

In order to discuss Delaunay-based meshing techniques, I must first briefly introduce the Delaunay tessellation itself. Discovered by Boris Delaunay in the 1930's, the Delaunay triangulation [17] is an optimal topological structure for the triangulation of points in the plane. Generalisations to higher-dimensions are possible, with a d-dimensional Delaunay

Figure 1.1: Example meshes for an airfoil geometry, illustrating the behaviour of the quadtree, advancing-front and Delaunay-refinement meshing strategies. In (i)-(iii) an axis-aligned quadtree-overlay is constructed, triangulated and warped to conform to the geometry. In (iv)-(vi) a conventional advancing-front technique *marches* the mesh outward from the boundaries, incrementally positioning new vertices until the fronts coalesce. In (vii)-(ix) Ruppert's Delaunay-refinement algorithm iteratively refines a coarse mesh of the domain until all element shape and size constraints are satisfied. Meshes in all cases were generated using equivalent input parameters.



structure tessellating points in $\mathbb{R}^d$ as a collection of d-simplex elements. The Delaunay tessellation is used extensively throughout this dissertation and its formal definition and a discussion of its properties is presented in Chapter 2. At this stage it suffices to know that the Delaunay tessellation is imbued with a number of convenient theoretical attributes and that, especially in $\mathbb{R}^2$, this type of tessellation is provably *optimal*.

Delaunay-refinement based methods for mesh generation typically start with the construction of a *constrained* Delaunay tessellation that conforms to the geometry of the domain to be meshed. New vertices are incrementally added to the interior of the domain until a suitable triangulation is formed. Vertices are added via a *refinement* strategy, in which their position is carefully chosen to eliminate elements that do not conform to a series of shape and/or size constraints. Throughout this refinement process, the topology of the underlying Delaunay tessellation is also updated, ensuring that the new mesh is also always a Delaunay tessellation. The method terminates when the set of constraints for all elements are satisfied. See Figure 1.1 for an illustration of the Delaunay-refinement process.

Delaunay-based methods for mesh generation facilitate the development of robust and provably-good meshing algorithms, in which the strong theoretical guarantees deriving from the properties of the Delaunay tessellation itself ensure that a number of worst-case element shape constraints can be satisfied, even for pathologically difficult inputs that may cause issues for heuristic methods like the advancing-front technique. Despite these theoretical assurances, for well-posed inputs, it is known that frontal methods typically outperform Delaunay-based techniques in terms of element shape quality and output size. Delaunay-based meshing techniques have become popular over the past decade, both with the computational geometers who study their theoretical properties, and with computational researchers and numericists drawn to their rich mathematical robustness. Early work on provably-good Delaunay based meshing was introduced by Chew, with his first refinement algorithm [14], dealing with uniform meshes for polygonal domains. Ruppert [43, 44] presented the first Delaunay-refinement scheme that ensured provably good shape- and size-optimality for general planar meshes, achieving angle bounds of $20.7^\circ \leq \theta(\tau) \leq 138.6^\circ$ for polygonal domains without *small* acute angles. Variations on his algorithm are still widely used today. Subsequent developments in Delaunay-based meshing were realised by a number of authors, with Shewchuk [50, 51], Cheng, Dey, Ramos, Levine [10, 11, 12], Edelsbrunner and Shah [19], Boissonnat and Oudot [5, 6], and Oudot, Rineau and Yvinec [39], amongst others, responsible for a number of innovations, including generalisations to surface and volumetric meshing in three-dimensions. A significant portion of this work is summarised in [13]. Further detail is provided in Chapters 2, 3, 4 and 5, where additional references are given.

1.2.4 Frontal-Delaunay Schemes

I also briefly mention the Frontal-Delaunay meshing technique, which is a hybridisation of the advancing-front and Delaunay-based strategies discussed previously. Frontal-Delaunay methods insert new vertices into the interior of the domain in a manner con-

sistent with the advancing-front technique. On the other hand, the topology of the mesh is maintained via a Delaunay tessellation – incrementally updated as new vertices are inserted. The resulting methods exhibit many of the good qualities of the advancing-front and Delaunay-based methods they derive from, inheriting both the high-quality practical performance of frontal methods and the robust theoretical framework of Delaunay techniques. These hybrid methods were initially introduced by Rebay [41] and Mavriplis [35] and have been studied by a range of authors, including Frey, Borouchaki and George [25], Remacle, Henrotte, Carrier-Baudouin, Béchet, Marchandise, Geuzaine and Mouton [42] and Erten and Üngör [20, 54]. Frontal-Delaunay methods form a significant part of this dissertation and thus a detailed discussion of their characteristics is deferred until Chapters 3, 4 and 5.

1.2.5 Mesh Improvement Procedures

In addition to techniques for genuine mesh construction, mesh improvement strategies, designed to improve the quality of existing meshes, are an important addition to the broader repertoire of meshing algorithms. Irrespective of the choices made concerning the primary method of mesh construction, almost all meshes can be enhanced, sometimes significantly, through the application of subsequent improvement techniques. Algorithms for mesh improvement seek to enhance the shape and/or size optimality of a given mesh, typically through the use of one or more of the following three strategies: (i) the iterative relocation of mesh vertices, (ii) the transformation of the underlying mesh topology and (iii) the insertion or removal of mesh vertices. Methods for mesh improvement based on vertex relocation operations, commonly referred to as *smoothing* techniques, include the well-known Laplacian smoothing method [21], schemes based on local optimisation, such as the methods of Freitag, Knupp, Jones, Plassman and Olivier-Gooch [22, 23, 24] and a range of methods constructed on the so-called *dual-mesh*, such as Centroidal Voronoi Tessellation (CVT) [9, 18] and Optimal Delaunay Tessellation (ODT) [8]. Laplacian smoothing is known to impose very low computational cost, but is generally outperformed in terms of quality by many of the other more sophisticated techniques, especially for higher-dimensional problems. Smoothing operations are often complemented through the addition of topological transformations to the mesh improvement scheme. Typically, these transformation are applied iteratively to local patches of elements, replacing the original element configurations with one that improves local mesh quality. While there exists a simple set of topological transformations for two-dimensional simplicial meshes, research on strategies for topological improvement for non-simplicial and/or higher-dimensional simplicial meshes is ongoing and includes, for example, the work of Rypl [47] and Joe [29] for mixed triangular-quadrilateral meshes and a series of techniques introduced by Briére de L'isle and George [7], de Cougny and Shephard [16], Misztal, Bærentzen, Anton and Erleben [36] and Joe [28] for tetrahedral meshes. Klinger and Shewchuk [30] present a state-of-the-art mesh improvement scheme for tetrahedral meshes, including techniques for optimal vertex insertion and removal. I return to methods for mesh improvement in Chapter 6, where additional discussions and references are included.

1.3 Objectives and Outline

The primary objective of this dissertation is to develop a set of high-quality meshing algorithms designed to tackle the type of complex geometrical problems often encountered when performing large-scale computational modelling and simulation studies.

In Chapter 2, I review a number of key geometrical structures and theorems from areas in computational geometry, with a focus on various spatial tessellations, including Delaunay tessellations and hierarchical methods for spatial decomposition and search. In addition to a discussion of theoretical properties, I present and review a number of practically efficient algorithms for the construction and maintenance of these geometrical objects.

In Chapters 3, 4 and 5, I focus on the development of methods for simplicial mesh generation in two- and three-dimensional spaces. Building upon previous research, I develop a new Frontal-Delaunay meshing strategy designed for planar, surface and volumetric problems. I show that this new approach typically outperforms the conventional Delaunay-refinement technique in terms of element quality, often achieving output competitive with advancing-front methods. I also demonstrate experimentally that the new algorithms achieve element shape- and size-bounds consistent with those derived for existing Delaunay-refinement schemes. In addition to this new Frontal-Delaunay paradigm, I investigate the use of alternative refinement strategies for volumetric meshing, designed to eliminate the occurrence of low-quality *sliver* tetrahedrons. Results show that the new Frontal-Delaunay method is particularly effective for planar and surface meshing problems, and that, when combined with alternative refinement strategies, leads to improved algorithms for the generation of volumetric meshes.

I address the question of mesh improvement in Chapter 6 and develop an optimisation program for the planar, surface and volumetric meshes generated using the Frontal-Delaunay algorithms presented in Chapters 3–5. My approach is based on a combination of optimisation-based vertex smoothing, generalised topological transformation operations and vertex insertion. I introduce a new priority-based schedule for mesh improvement and demonstrate experimentally that the new algorithm is an effective tool for the improvement of two- and three-dimensional simplicial meshes. Specifically, results show that the new technique often significantly outperforms existing state-of-the-art approaches in terms of computational efficiency while achieving competitive levels of optimisation quality.

I offer a final summary of results and conclusions in Chapter 7, focusing also on a number of objectives for future research.

References

[1] Alliez, P., Cohen-Steiner, D., Yvinec, M., and Desbrun, M. Variational Tetrahedral Meshing. *ACM Transactions on Graphics 24*, 3 (2005), 617–625.

[2] Babuška, I., and Aziz, A. K. On the Angle Condition in the Finite Element Method. *SIAM J. Numer. Anal. 13* (Apr. 1976), 214–226.

[3] Bern, M., Eppstein, D., and Gilbert, J. Provably Good Mesh Generation. *J. Comput. Syst. Sci 48* (1990), 231–241.

[4] Blacker, T. D., and Stephenson, M. B. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *International Journal for Numerical Methods in Engineering 32*, 4 (1991), 811–847.

[5] Boissonnat, J. D., and Oudot, S. Provably Good Surface Sampling and Approximation. In *ACM International Conference Proceeding Series* (2003), vol. 43, pp. 9–18.

[6] Boissonnat, J. D., and Oudot, S. Provably Good Sampling and Meshing of Surfaces. *Graphical Models 67*, 5 (2005), 405–451.

[7] Briére de L'isle, E., and George, P. L. Optimization of tetrahedral meshes. In *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, I. Babuska, W. D. Henshaw, J. E. Oliger, J. E. Flaherty, J. E. Hopcroft, and T. Tezduyar, Eds., vol. 75 of *The IMA Volumes in Mathematics and its Applications*. Springer New York, 1995, pp. 97–127.

[8] Chen, L., and Holst, M. Efficient mesh optimization schemes based on optimal Delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering 200*, 9 (2011), 967–984.

[9] Chen, Z., Wang, W., Lévy, B., Liu, L., and Sun, F. Revisiting Optimal Delaunay Triangulation for 3D Graded Mesh Generation. *SIAM Journal on Scientific Computing 36*, 3 (2014), A930–A954.

[10] Cheng, S., and Dey, T. Quality Meshing with Weighted Delaunay Refinement. *SIAM Journal on Computing 33*, 1 (2003), 69–93.

[11] Cheng, S. W., Dey, T. K., and Levine, J. A. A Practical Delaunay Meshing Algorithm for a Large Class of Domains*. In *Proceedings of the 16th international meshing roundtable* (2008), Springer, pp. 477–494.

[12] Cheng, S. W., Dey, T. K., and Ramos, E. A. Delaunay Refinement for Piecewise Smooth Complexes. *Discrete & Computational Geometry 43*, 1 (2010), 121–166.

[13] Cheng, S. W., Dey, T. K., and Shewchuk, J. R. *Delauay Mesh Generation*. Taylor & Francis, New York, 2013.

[14] Chew, L. P. Guaranteed-quality Triangular Meshes. Tech. rep., Cornell University, Department of Computer Science, Ithaca, New York, 1989.

[15] Courant, R., Friedrichs, K., and Lewy, H. ber die partiellen Differenzengleichungen der Mathematischen Physik. *Mathematische Annalen 100*, 1 (1928), 32–74.

[16] de Cougny, H. L., and Shephard, M. S. Refinement, Derefinement, and Optimization of Tetrahedral Geometric Triangulations in Three Dimensions. 1995.

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

[18] Du, Q., Faber, V., and Gunzburger, M. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review 41*, 4 (1999), 637–676.

[19] Edelsbrunner, H., and Shah, N. R. Triangulating Topological Spaces. *International Journal of Computational Geometry & Applications 7*, 04 (1997), 365–378.

[20] Erten, H., and Üngör, A. Quality Triangulations with Locally Optimal Steiner Points. *SIAM J. Sci. Comp. 31*, 3 (2009), 2103–2130.

[21] Field, D. A. Laplacian Smoothing and Delaunay Triangulations. *Communications in applied numerical methods 4*, 6 (1988), 709–712.

[22] Freitag, L., Jones, M., and Plassmann, P. A Parallel Algorithm for Mesh Smoothing. *SIAM Journal on Scientific Computing 20*, 6 (1999), 2023–2040.

[23] Freitag, L. A., and Knupp, P. M. Tetrahedral Mesh Improvement via Optimization of the Element Condition Number. *International Journal for Numerical Methods in Engineering 53*, 6 (2002), 1377–1391.

[24] Freitag, L. A., and Ollivier-Gooch, C. Tetrahedral Mesh Improvement using Swapping and Smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[25] Frey, P. J., Borouchaki, H., and George, P. L. 3D Delaunay Mesh Generation Coupled with an Advancing-Front Approach. *Computer Methods in Applied Mechanics and Engineering 157*, 12 (1998), 115 – 131.

[26] Fried, I. Condition of Finite Element Matrices Generated from Nonuniform Meshes. *AIAA Journal 10* (Feb. 1972), 219–221.

[27] George, J. A. *Computer Implementation of the Finite Element Method*. PhD thesis, 1971.

[28] Joe, B. Construction of Three-dimensional Improved-Quality Triangulations Using Local Transformations. *SIAM Journal on Scientific Computing 16*, 6 (1995), 1292–1307.

[29] Joe, B. Flips for Quadrilateral Meshes. Tech. rep., 2008.

[30] Klingner, B. M., and Shewchuk, J. R. Aggressive Tetrahedral Mesh Improvement. In *Proceedings of the 16th International Meshing Roundtable*, M. L. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 3–23.

[31] Křížek, M. On the Maximum Angle Condition for Linear Tetrahedral Elements. *SIAM J. Numer. Anal. 29*, 2 (Apr. 1992), 513–520.

[32] Lax, P., and Wendroff, B. Systems of Conservation Laws. *Communications on Pure and Applied Mathematics 13*, 2 (1960), 217–237.

[33] LeVeque, R. J. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[34] Löhner, R., and Parikh, P. Generation of Three-dimensional Unstructured Grids by the Advancing-Front Method. *International Journal for Numerical Methods in Fluids 8*, 10 (1988), 1135–1149.

[35] Mavriplis, D. J. An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *Journal of Computational Physics 117*, 1 (1995), 90 – 101.

[36] Misztal, M. K., Bærentzen, J. A., Anton, F., and Erleben, K. Tetrahedral Mesh Improvement Using Multi-face Retriangulation. In *Proceedings of the 18th International Meshing Roundtable*, B. W. Clark, Ed. Springer Berlin Heidelberg, 2009, pp. 539–555.

[37] Mitchell, S. A., and Vavasis, S. A. Quality Mesh Generation in Three-dimensions. In *Symposium on Computational Geometry* (1992), pp. 212–221.

[38] Mitchell, S. A., and Vavasis, S. A. Quality Mesh Generation in Higher-dimensions. *SIAM J. Comput. 29*, 4 (2000), 1334–1370.

[39] Oudot, S., Rineau, L., and Yvinec, M. Meshing Volumes Bounded by Smooth Surfaces. In *Proceedings of the 14th International Meshing Roundtable* (2005), Springer, pp. 203–219.

[40] Peraire, J., Peiró, J., and Morgan, K. Advancing-Front Grid Generation. In *Handbook of Grid Generation*, J. F. Thompson, B. K. Soni, and N. P. Weatherill, Eds. Taylor & Francis, 1998.

[41] Rebay, S. Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal of Computational Physics 106*, 1 (1993), 125 – 138.

[42] Remacle, J. F., Henrotte, F., Carrier-Baudouin, T., Béchet, E., Marchandise, E., Geuzaine, C., and Mouton, T. A Frontal-Delaunay Quad-Mesh Generator using the $\mathbb{L}^\infty$-norm. *International Journal for Numerical Methods in Engineering 94*, 5 (2013), 494–512.

[43] Ruppert, J. A New and Simple Algorithm for Quality 2-dimensional Mesh Generation. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms* (Philadelphia, PA, USA, 1993), SODA '93, Society for Industrial and Applied Mathematics, pp. 83–92.

[44] Ruppert, J. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms 18*, 3 (1995), 548 – 585.

[45] Rypl, D. *Sequential and Parallel Generation of Unstructured 3D Meshes.* PhD thesis, CTU in Prague, 1998.

[46] Rypl, D. Approaches to Discretization of 3D Surfaces. In *CTU Reports*, vol. 7. CTU Publishing House, Prague, Czech Republic, 2003.

[47] Rypl, D., and Bittnar, Z. Hybrid Method for Generation of Quadrilateral Meshes. *Engineering Mechanics 9*, 1/2 (2002), 49–64.

[48] Schöberl, J. NETGEN: An Advancing Front 2D/3D Mesh Generator based on Abstract Rules. *Computing and Visualization in Science 1*, 1 (1997), 41–52.

[49] Schreiner, J., Scheidegger, C. E., Fleishman, S., and Silva, C. T. Direct (Re)Meshing for Efficient Surface Processing. *Computer Graphics Forum 25*, 3 (2006), 527–536.

[50] Shewchuk, J. R. *Delaunay Refinement Mesh Generation.* PhD thesis, Pittsburg, Pennsylvania, 1997.

[51] Shewchuk, J. R. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry* (New York, NY, USA, 1998), SCG '98, ACM, pp. 86–95.

[52] Shewchuk, J. R. What is a Good Linear Element? – Interpolation, Conditioning, and Quality Measures. In *In 11th International Meshing Roundtable* (2002), pp. 115–126.

[53] Synge, J. L. *The Hypercircle in Mathematical Physics: A Method for the Approximate Solution of Boundary Value Problems.* Cambridge University Press, 2012.

[54] Üngör, A. Off-centers: A New Type of Steiner Points for Computing Size-optimal Guaranteed-quality Delaunay Triangulations. *Computational Geometry: Theory and Applications 42*, 2 (2009), 109–118.

[55] Yerry, M. A., and Shephard, M. S. A Modified Quadtree Approach to Finite Element Mesh Generation. *Computer Graphics and Applications, IEEE 3*, 1 (1983), 39–46.

[56] Yerry, M. A., and Shephard, M. S. Automatic Three-dimensional Mesh Generation by the Modified-octree Technique. *International Journal for Numerical Methods in Engineering 20*, 11 (1984), 1965–1990.

Chapter 2

Elements, Tessellations & Spatial Trees

Algorithms for mesh generation are built upon a number of important theorems and results from computational geometry. In this chapter I review the fundamental geometrical structures that will be used throughout the remainder of this dissertation, including simplicial elements, Delaunay tessellations and Voronoi diagrams. Additionally, I discuss a number of alternate techniques for spatial subdivision and search, including hierarchical decompositions and spatial trees. I present both an outline of the important theoretical properties of these structures and the development of efficient and practical algorithms for their construction and maintenance.

2.1 Preliminaries

I begin by reviewing a number of fundamental constructs and definitions, adapted from the comprehensive presentations developed by Cheng, Dey and Shewchuk in [15, Chapter 1]:

Definition 2.1 (affine hull)**.** Given a set of points $X \subset \mathbb{R}^d$, an *affine combination* of the points in X is a point $\mathbf{p}$ given by the linear combination $\mathbf{p} = \sum_{i=1}^{n} w_i \, \mathbf{x}_i$, such that the scalar weights satisfy $\sum_{i=1}^{n} w_i = 1$. The *affine hull* of X, denoted $\mathrm{Aff}(X)$, is the set containing all affine combinations of the points in X.

Given a set of points X, a point $\mathbf{p}$ is said to be *affinely independent* of X if it cannot be written as an affine combination of the points in X, which implies $\mathbf{p} \notin \mathrm{Aff}(X)$. By extension, the set X is said to be affinely independent if all points $\mathbf{x}_i \in X$ are affinely independent of the others. In $\mathbb{R}^d$ at most $d+1$ points can be affinely independent – sets containing more than $d+1$ points lie on a common hyperplane.

Definition 2.2 (convex hull)**.** Given a set of points $X \subset \mathbb{R}^d$, a *convex combination* of the points in X is a point $\mathbf{p}$ that can be written as a non-negative affine combination, requiring $w_i \geq 0$ for all i. The *convex hull* of X, denoted $\mathrm{Conv}(X)$, is the set containing all convex combinations of the points in X, such that $\mathrm{Conv}(X) = \left\{ \mathrm{Aff}(X) \,\middle|\, \forall i : w_i \geq 0 \right\}$

Geometrically, convex hulls can be visualised in terms of a 'rubber-band' analogy. Given a set of points X in $\mathbb{R}^2$, $\text{Conv}(X)$ is the polygon formed by an elastic band that is stretched to minimally enclose all points in X. This analogy can be extended to higher dimensions by considering enclosing elastic hyper-membranes. Clearly, when a set X in $\mathbb{R}^d$ contains more than $d+1$ points it is not required that all points lie on the boundary of the convex hull. Points that lie on the boundary of $\text{Conv}(X)$ are said to be *vertices* of the hull. Sets containing exactly $d+1$ affinely independent points support *unique* convex hulls.

Definition 2.3 (simplex)**.** A *d-simplex* is the convex hull of a set of $d+1$ affinely independent points in $\mathbb{R}^d$.

The d-simplex is a non-degenerate convex polyhedron in $\mathbb{R}^d$ containing a minimal number of vertices. Based on the properties of convex hulls discussed previously, it is clear that such polyhedrons contain the minimal $d+1$ affinely independent vertices required to support a full-dimensional hull in $\mathbb{R}^d$. In such cases, the resulting hull defines a unique convex polyhedron. Specifically, a 0-simplex is a vertex, a 1-simplex is an edge, a 2-simplex is a triangle and a 3-simplex is a tetrahedron. Accordingly, it can be seen that the boundary of a general d-simplex can be decomposed into a hierarchy of lower dimensional simplexes, or *faces*, from dimension 0 to $d-1$, with, for example, the boundary of a tetrahedron consisting of a collection of triangles, edges and vertices. Specifically, given a simplex τ, the 0-faces of τ are its vertices, the 1-faces are its edges, and so-on. In particular the $(d-1)$-faces of a simplex are called its *facets*. A d-simplex has $d+1$ facets. In this study, I refer to 2- and 3-simplexes as triangular and tetrahedral *elements*. These elements are the foundation of the meshing algorithms developed in subsequent chapters.

Definition 2.4 (simplicial complex)**.** A *simplicial complex* $\mathcal{T}$ is a finite set of simplexes, satisfying the following constraints:

(i) $\mathcal{T}$ contains the faces of all simplexes $t_i \in \mathcal{T}$.

(ii) For all pairs of simplexes $\tau_i, \tau_j \in \mathcal{T}$, their intersection $\tau_i \cap \tau_j$ is either empty, or a face common to both τ_i and τ_j.

Geometrically, simplicial complexes can be visualised as structures created by 'gluing' collections of simplexes together. Two simplexes in a complex are said to be *adjacent* if they share a common face, and *disjoint* otherwise. The second constraint in Definition 2.4 prohibits collections of simplexes that intersect *internally*, requiring that adjacent simplexes intersect at a vertex (0-face), along a full-edge (1-face) or over a higher dimensional simplicial face. The meshing algorithms developed throughout this dissertation are based on so-called *homogeneous* simplicial d-complexes, in which, given a simplicial complex $\mathcal{P}$, all simplexes $f \in \mathcal{P}$ of dimension 0 through $d-1$ are faces of a d-simplex $\tau \in \mathcal{P}$. Specifically, simplicial 2- and 3-complexes are conforming meshes of triangular and tetrahedral elements, respectively. Throughout this work, I use the terminology interchangeably. The notion of simplicial complexes can be extended to encapsulate various other geometric structures, including *polyhedral complexes*, *piecewise linear complexes* and *piecewise*

smooth complexes. These complexes extend Definition 2.4 by considering collections of non-simplicial elements, known more generally as the *linear-cells* of a complex, and will be revisited in later chapters.

Definition 2.5 (underlying space)**.** The *underlying space* of a complex $\mathcal{P}$, denoted $|\mathcal{P}|$, is the union of its cells, $|\mathcal{P}| = \bigcup_{C \in \mathcal{P}} C$.

The underlying space of a complex $\mathcal{P}$ is a formal definition of what is considered to lie *inside* of $\mathcal{P}$. For example, given a general point $\mathbf{x}$, such that $\mathbf{x} \in |\mathcal{P}|$, it is known that the point $\mathbf{x}$ lies in the complex $\mathcal{P}$.

2.1.1 Simplex Shape

Before moving on to discussions of the Delaunay tessellation itself, it's necessary to first introduce a number of constructs used to measure the size, shape and quality of the individual simplexes that make up a tessellation.

Definition 2.6 (Euclidean ball)**.** In $\mathbb{R}^d$ the Euclidean ball, or *closed d-ball*, of radius $r > 0$ and centre $\mathbf{c}$, is $\mathrm{B}(\mathbf{c}, r) = \left\{ \mathbf{x} \in \mathbb{R}^d \,\middle|\, \|\mathbf{x} - \mathbf{c}\| \leq r \right\}$, where $\|\mathbf{p} - \mathbf{q}\|$ is simply the pairwise Euclidean distance between the points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d$. The *boundary* of the d-ball is called a $(d-1)$-*sphere* and is the subset of $\mathrm{B}(\mathbf{c}, r)$ such that $\|\mathbf{x} - \mathbf{c}\| = r$.

Geometrically, d-balls and d-spheres are simply generalisations of conventional spherical structures to higher dimensional spaces. The 1-sphere is a circle and the 2-sphere is simply the 'conventional' sphere in $\mathbb{R}^3$. Additionally, a d-ball is either *closed* or *open*, depending on whether or not the ball is inclusive of its boundary.

Definition 2.7 (circumball)**.** Given a simplex τ embedded in $\mathbb{R}^d$, the *circumball* of τ is any d-ball whose boundary passes through all vertices of τ exactly. The boundary of a circumball is a $(d-1)$-sphere called a *circumsphere*.

If a simplex has the same dimension as the space in which it is embedded, its associated circumball is unique. *Full-dimensional* simplexes, such as triangles in $\mathbb{R}^2$ and tetrahedra in $\mathbb{R}^3$, are associated with unique circumballs as a consequence. If, on the other hand, a simplex is embedded in a higher dimensional space, this uniqueness is lost, with an infinite family of circumballs associated with the simplex instead. Consider, for example, the common case of a triangle embedded in $\mathbb{R}^3$. In this case, a family of circumballs can be placed along a vector normal to the plane of the element. For cases such as this, the ambiguity can be resolved by selecting the *smallest* circumball from the set, which is known as its *diametric-ball*. See the extended exposition presented by Cheng, Dey and Shewchuk [15, Chapter 1] for additional discussions and proofs.

Definition 2.8 (diametric ball)**.** Given a simplex τ embedded in $\mathbb{R}^d$, the *diametric-ball* of τ is the associated circumball of minimum radius. The centre of the diametric-ball is called the *circumcentre* of τ and lies on the affine hull of the simplex τ.

Revisiting the case of a triangle embedded in $\mathbb{R}^3$, its diametric-ball is the circumball whose centre lies in the plane of the triangle. The significance of the circumcentre will

be discussed extensively in the context of the Delaunay meshing algorithms presented in later chapters.

Definition 2.9 (min-enclosing ball)**.** Given a simplex τ embedded in $\mathbb{R}^d$, the *min-enclosing ball* of τ is the d-ball of minimum radius that encloses all vertices in τ.

The min-enclosing ball for a d-simplex τ is always the diametric ball of a face of τ. When the min-enclosing ball is equal to the diametric ball of a lower dimensional face it is important to note that the min-enclosing ball is smaller than the diametric ball of τ.

2.1.2 Simplex Quality

An understanding of the *shape-quality* of simplicial elements is fundamental to the development of the high-quality meshing algorithms pursued throughout this work. As per the discussion presented in Section 1.1, it is known that high-quality simplicial elements should not contain overly small or large angles and, additionally, should not be highly skewed or distorted. Taking an *isotropic* definition, simplexes of ideal quality are those that are perfectly *regular* – consisting of equilateral triangles and tetrahedrons for the two- and three-dimensional problems considered in this study. These elements can be distorted into *poor*-quality shapes in a range of different ways. See Figures 2.1 and 2.2 for a catalogue of the various poor-quality geometric configurations that can be achieved by 2- and 3-simplexes.

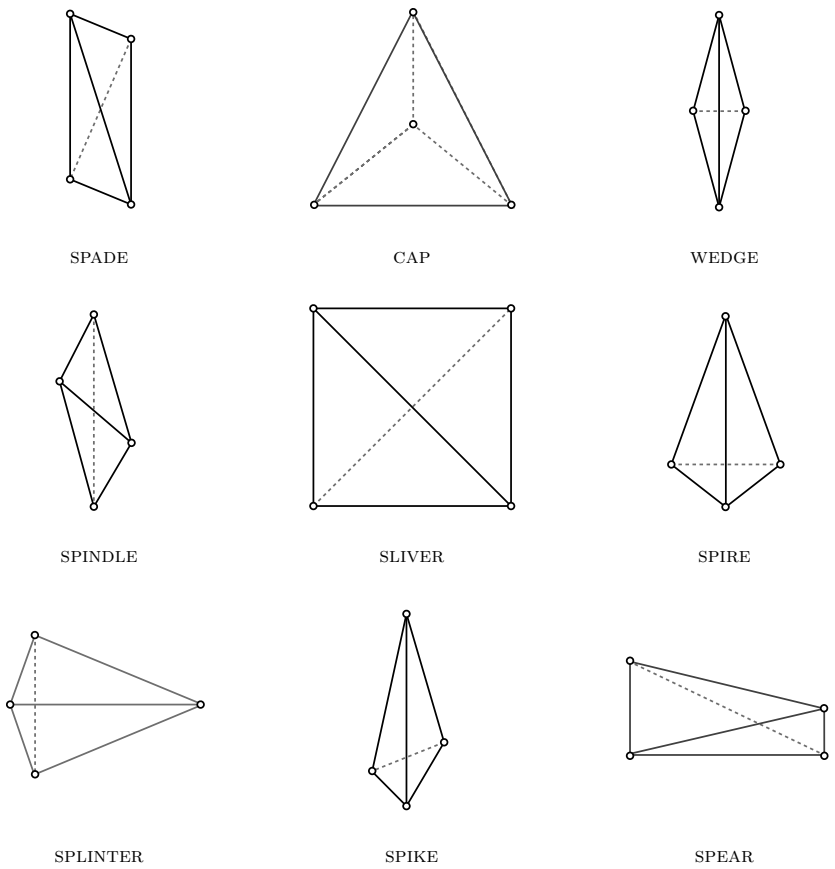
Element quality is typically defined in terms of a scalar *quality-function* that assigns simplexes a normalised *score* based on their geometrical configuration. Possibly the simplest and most intuitive measure of the quality of a given simplex is based on a consideration of its angles. The shape quality of a triangular element can be characterised in terms of the *plane angles* between adjacent edges, while the quality of tetrahedral elements requires consideration of both the *dihedral angles* between adjacent triangular faces and the plane angles between the adjacent edges on each triangular face. Importantly, note that some of the the poor-quality tetrahedral geometries illustrated in Figure 2.2 achieve good distributions for either dihedral or plane angles, such as the *sliver*, which has good plane angles, or the *spear*, which has good dihedral angles. It should be noted that no poor-quality tetrahedron achieves a good distribution for both plane and dihedral angles.

Despite their intuitive appeal, angle-based quality measures are inherently multi-valued, consisting of three plane angles per 2-simplex and six dihedral plus twelve plane angles per 3-simplex. While a number of strategies can be adopted to extract a single-valued score, such as weighted sums or considerations of minimum and maximum values [26], the high computational cost of these measures is unattractive in practice. The first genuinely single-valued simplicial quality measure introduced is the so-called *radius-edge* ratio, which is widely used in the context of Delaunay-based meshing algorithms.

Definition 2.10 (radius-edge ratio)**.** The *radius-edge* ratio $\rho(\tau)$ of a d-simplex τ is $R/\|\mathbf{e}_{\min}\|$, where R is the radius of the diametric ball of τ and $\|\mathbf{e}_{\min}\|$ is the length of its shortest edge.

Figure 2.1: Poor quality element configurations for 2-simplex elements, showing the NEEDLE and CAP types, respectively.

Figure 2.2: Poor quality element configurations for 3-simplex elements, showing the SPADE, CAP, WEDGE, SLIVER, SPINDLE, SPIRE, SPLINTER, SPIKE and SPEAR types, respectively.



In $\mathbb{R}^2$, the radius-edge ratio is an ideal measure of element quality, detecting all poor-quality triangular configurations illustrated in Figure 2.1. Unfortunately, in $\mathbb{R}^3$, this optimality is compromised, with the radius-edge ratio failing to detect the *sliver* configuration, in which all edge lengths are relatively large, despite the pathologically poor dihedral angles. The radius-edge ratio is still able to detect the remaining poor-quality tetrahedral configurations presented in Figure 2.2. The radius-edge ratio achieves a minimum for equilateral elements – $1/\sqrt{3}$ for triangles and $\sqrt{6}/4$ for tetrahedrons, increasing toward $+\infty$ as elements tend toward degeneracy. For triangles, the radius-edge ratio is robust and can be related to the minimum plane angle $\theta_{\min}$, such that $\rho(\tau) = R/\|\mathbf{e}_{\min}\| = \frac{1}{2}(\sin(\theta_{\min}))^{-1}$. Due to the summation of angles in a triangle, given a minimum angle $\theta_{\min}$ the largest angle $\theta_{\max}$ is clearly bounded, such that $\theta_{\max} \leq 180^\circ - 2\,\theta_{\min}$. Importantly, note that there is no relationship between the radius-edge ratio and the dihedral angles of tetrahedral elements. These relationships will be revisited in later chapters.

The lack of robustness associated with the radius-edge measure for higher-dimensional problems has motivated the search for alternate formulations of element quality. First introduced by Parthasarathy, Graichen and Hathaway [36], the so-called *volume-length* measure is an alternative definition of simplex quality that is known to detect all poor-quality configurations for both triangular and tetrahedral elements.

Definition 2.11 (volume-length ratio)**.** The *volume-length* ratio $v(\tau)$ of a d-simplex τ is $V/\|\mathbf{e}_{\text{rms}}\|^d$, where V is the signed volume of τ and $\|\mathbf{e}_{\text{rms}}\|$ is the root-mean-square of its edge lengths.

Typically, the volume-length ratio $v(\tau)$ is normalised to achieve a score of $+1$ for ideal elements. As an element distorts toward degeneracy $v(\tau) \to 0$, with fully inverted elements achieving a score of -1. For triangles the 'volume' is simply taken as the signed area of the element. Due to its robustness and simplicity, the volume-length measure is widely used for a variety of meshing applications. Specifically, in [30], Klinger and Shewchuk show that it is the best all-round quality metric for their state-of-the-art tetrahedral mesh improvement program. This measure is used extensively throughout subsequent sections of this thesis.

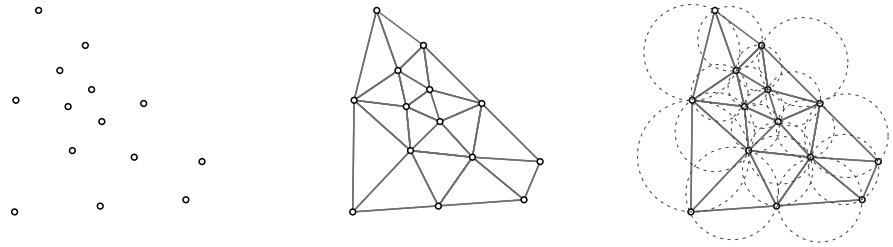
2.2 Delaunay Tessellations

The Delaunay tessellation, introduced by Boris Delaunay in 1934 [17], is one of the most widely used geometrical structures in computational geometry and underpins much of the work developed in the later chapters of this study.

Definition 2.12 (Simplicial tessellation)**.** Given a finite set of points $X \subset \mathbb{R}^d$, a *simplicial tessellation* of X is a simplicial complex $\mathcal{T}$, where the points in X are the vertices of $\mathcal{T}$ and $\text{Conv}(X) = \bigcup_{\tau_i \in \mathcal{T}} \tau_i$.

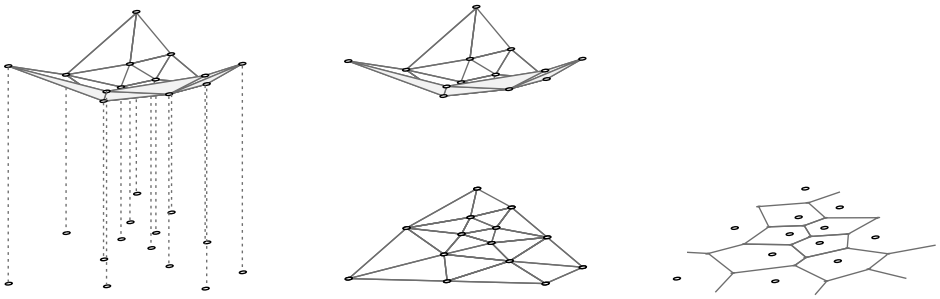
Definition 2.13 (Delaunay tessellation)**.** Given a finite set of points $X \subset \mathbb{R}^d$, the *Delaunay tessellation* of X, denoted $\text{Del}(X)$, is a simplicial tessellation of X such that all simplexes in $\text{Del}(X)$ are *Delaunay*. A simplex τ is said to be Delaunay if the intersection

Figure 2.3: An example of the Delaunay triangulation $\mathrm{Del}(X)$ for a set of points $X \subset \mathbb{R}^2$, showing: (i) the distribution of points X, (ii) the topology of the Delaunay triangulation $\mathrm{Del}(X)$, and (iii) the empty circumcircles associated with the triangles $\tau \in \mathrm{Del}(X)$.



(i) (ii) (iii)

Figure 2.4: Illustration of the links between Delaunay tessellations, Voronoi diagrams and convex hulls for vertices $X \subset \mathbb{R}^2$, showing (i) *lifting* the points X onto $X^+ \in \mathbb{R}^3$ along with the associated lower hull $\mathrm{Conv}\big(X^+\big)$, (ii) the Delaunay triangulation $\mathrm{Del}(X)$ as the projection of $\mathrm{Conv}\big(X^+\big)$ onto $\mathbb{R}^2$, and (iii) the dual Voronoi diagram $\mathrm{Vor}(X)$ associated with $\mathrm{Del}(X)$.



(i) (ii) (iii)

of its *open* diametric ball and the points in X is *empty*. A simplex τ is said to be *strongly Delaunay* if the intersection of its *closed* diametric ball and X contains the vertices $\mathbf{x} \in \tau$ only.

The Delaunay tessellation has been widely studied in the literature and I do not intend to review its properties in full here. I refer the reader to, for example, the comprehensive overview presented by Cheng, Dey and Shewchuk in [15, Chapters 2–5] for additional details. In the context of this study, it is sufficient to note that a Delaunay tessellation $\mathrm{Del}(X)$ can be constructed for any finite set $X \subset \mathbb{R}^d$ and that in cases where all simplexes in $\mathrm{Del}(X)$ are *strongly* Delaunay, such a tessellation is unique. So-called *co-spherical* point-sets – those with more than $d+1$ points lying on the boundary of a common d-ball – do not support tessellations that are strongly Delaunay. Though associated Delaunay tessellations still exists for these *degenerate* inputs, elements that *span* the co-spherical vertex subsets are not unique. In practice, algorithms for the construction of Delaunay tessellations typically deal with these degenerate vertex distributions by simply choosing one of the non-unique element combinations available, such that a conforming tessellation is achieved. Many highly structured problems include co-spherical vertex configurations, such as, for example, those based on uniform Cartesian grids.

The Delaunay tessellation possesses a number of desirable theoretical properties. Amongst the collection of all possible triangulations for a set of points $X \subset \mathbb{R}^2$, the Delaunay triangulation $\text{Del}(X)$ is known to simultaneously maximise the minimum plane angle $\theta_{\min}$ whilst also minimising the largest circumdisk and min-enclosing disk for all simplexes $\tau \in \text{Del}(X)$. Unfortunately, in higher dimensions, much of this optimally is lost, with higher dimensional Delaunay tessellations only guaranteeing that a minimisation of the largest min-enclosing ball is achieved. The absence of bounds on the dihedral angle for higher dimensional cases is a serious impediment to the development of volumetric meshing algorithms of provable quality, and is an issue that will be revisited extensively in later chapters. A simple example of Delaunay tessellation in the plane is shown in Figure 2.3, illustrating the empty circumdisk property.

The Delaunay tessellation is also closely related to a number of other important geometrical structures, namely the convex hull and the *Voronoi* diagram.

Definition 2.14 (Voronoi region)**.** Given a finite set of sites $X \subset \mathbb{R}^d$, the *Voronoi region* v_i associated with a site $\mathbf{x}_i \in X$ is the set of all points in $\mathbb{R}^d$ lying at least as close to $\mathbf{x}_i$ as any other site $\mathbf{x}_j \in X$, $\forall j \neq i$.

The Voronoi regions v_i are convex polyhedrons, consisting of a hierarchy of Voronoi vertices (0-faces), Voronoi edges (1-faces), Voronoi facets ($(d-1)$-faces) and Voronoi regions (d-faces). Importantly, all Voronoi faces are convex, and all faces, other than the vertices, can be unbounded.

Definition 2.15 (Voronoi diagram)**.** Given a finite set of points $X \subset \mathbb{R}^d$, the *Voronoi diagram* $\text{Vor}(X)$ is a polyhedral complex containing the set of Voronoi regions v_i associated with the sites $\mathbf{x}_i \in X$.

Geometrically, the Delaunay tessellation and Voronoi diagram are said to be geometric *duals*. Given a set of points $X \subset \mathbb{R}^d$, the edges (1-faces) of the Voronoi regions $v_i \in \text{Vor}(X)$ are equivalent to the line segments formed by joining the circumcentres associated with pairs of facet-adjacent simplexes in the Delaunay tessellation $\text{Del}(X)$. These edges are orthogonal to the common $(d-1)$-facets shared between the adjacent simplexes $\tau_i, \tau_j \in \text{Del}(X)$. It is important to note that the Voronoi diagram is essentially a representation of the Euclidean distance function associated with a set of points. Specifically, given a set $X \subset \mathbb{R}^d$, the edges and vertices of the Voronoi diagram $\text{Vor}(X)$ correspond to local ridges and peaks associated with the closest distance map of X. This relationship implies that points positioned faces in $\text{Vor}(X)$ are, in some sense, locally 'well-separated' with respect to the points X. A number of the meshing algorithms introduced in the later chapters of this study are based on these observations.

Additionally, it is well known that the Delaunay tessellation of a set of points $X \subset \mathbb{R}^d$ is related to the convex hull of the associated set X^+, where X^+ is created by *lifting* the original points X into $\mathbb{R}^{d+1}$. This so-called lifting is achieved via a *parabolic map* that associates each point $\mathbf{x}_i \in X$ with its *lifted companion* $\mathbf{x}_i^+ \in X^+$, such that $\mathbf{x}_i^+ = \left(c_1, c_2, ..., c_d, c_1^2 + c_2^2 + ... + c_d^2\right)$, where the c_i's are the coordinates of the points $\mathbf{x}_i \in X$. The Delaunay tessellation $\text{Del}(X)$ can be recovered by projecting the *downward-facing*

facets of $\text{Conv}(X^+)$ back onto $\mathbb{R}^d$. See Figure 2.4 for an illustration of the links between Delaunay tessellations, Voronoi diagrams and convex hulls.

2.2.1 Delaunay Tessellations with Constraints

While the conventional Delaunay tessellation is a useful geometric structure in its own right, its application to problems in mesh generation requires the consideration of additional geometric constraints. In the context of meshing algorithms, it is clear that a tessellation is required to *conform* to the geometry of the domain to be meshed.

Definition 2.16 (conformity)**.** A complex $\mathcal{T}$ is said to *conform* to a complex $\mathcal{P}$ if $|\mathcal{T}| = |\mathcal{P}|$ and every cell in $\mathcal{P}$ can be written as a union of cells in $\mathcal{T}$.

Firstly, it is typical to consider the triangulation of linear geometrical objects – those made up of collections of lines, polygons and polyhedrons in $\mathbb{R}^2$ and $\mathbb{R}^3$. These types of domains are encapsulated by a generalised linear geometrical structure, known as a piecewise linear complex.

Definition 2.17 (piecewise linear complex)**.** A *piecewise linear complex* (PLC) $\mathcal{P}$ is a finite set of linear cells satisfying the following conditions:

(i) The vertices and edges in $\mathcal{P}$ form a simplicial complex.

(ii) The boundary of each linear cell $C \in \mathcal{P}$ is a union of linear cells in $\mathcal{P}$.

(iii) Given two distinct linear cells $C_i, C_j \in \mathcal{P}$, their intersection $C_i \cap C_j$ is either empty or a union of lower dimensional linear cells in $\mathcal{P}$.

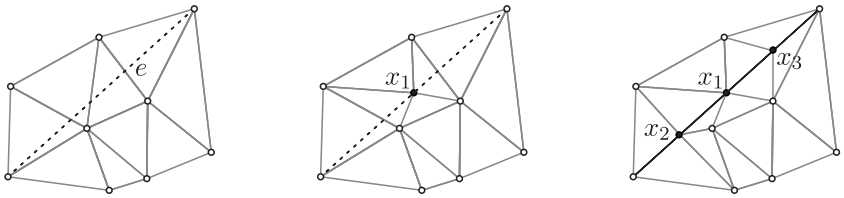
Given a general PLC $\mathcal{P}$, it is natural to consider whether a suitable simplicial tessellation can be constructed, such that all geometric constraints are satisfied. Such a tessellation is known as a *triangulation* of $\mathcal{P}$, and can be considered both with and without the inclusion of additional vertices.

Definition 2.18 (triangulation of a PLC)**.** Let $\mathcal{P}$ be a general PLC. A simplicial complex $\mathcal{T}$ is called a *triangulation* of $\mathcal{P}$ if $\mathcal{T}$ conforms to $\mathcal{P}$ and if $\mathcal{T}$ and $\mathcal{P}$ contain the same vertices.

Definition 2.19 (Steiner triangulation of a PLC)**.** Let $\mathcal{P}$ be a general PLC. A simplicial complex $\mathcal{T}$ is called a *Steiner triangulation* of $\mathcal{P}$ if $\mathcal{T}$ conforms to $\mathcal{P}$ and if $\mathcal{T}$ contains all vertices in $\mathcal{P}$. Additionally vertices in $\mathcal{T}$ are called *Steiner points*.

It is known that every PLC $\mathcal{P}$ in $\mathbb{R}^2$ supports a triangulation $\mathcal{T}$, commonly also referred to as a *conforming triangulation* of $\mathcal{P}$. Unfortunately, the same is not true in higher dimensions; specifically, in $\mathbb{R}^3$, it is known that certain PLC's exist that cannot be triangulated without the inclusion of additional vertices. A common, and disarmingly simple example, is the so-called Schönhardt polyhedron – a 'twisted' triangular prism that cannot be decomposed into a conforming tetrahedral complex. The meshing process in higher dimensions is significantly complicated by the existence of PLC's that cannot be triangulated.

Figure 2.5: Use of a Steiner Delaunay triangulation to satisfy geometric constraints. In (i) the original triangulation $\text{Del}(X)$ is shown with an unsatisfied edge constraint e. Additional Steiner points are added to $\text{Del}(X)$ through recursive bisection in (ii)-(iii) until the constraint is recovered such that $e = \bigcup_{f \in \text{Del}(X)} f$.

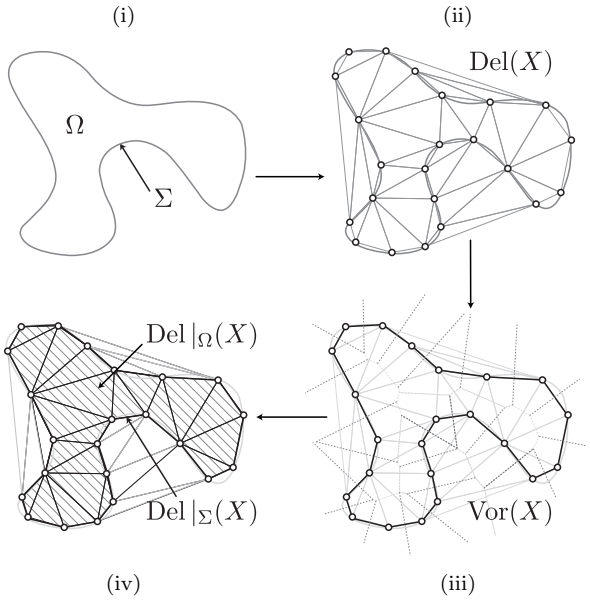


(i) (ii) (iii)

In the context of mesh generation, additional vertices, known as *Steiner points* are typically added to a tessellation, resulting in a so-called Steiner triangulation $\mathcal{T}$ of a PLC $\mathcal{P}$. If each simplex $\tau \in \mathcal{T}$ is also Delaunay, such a triangulation is known as a *Steiner Delaunay triangulation* of $\mathcal{P}$. In practice, these conforming Delaunay triangulations are created in a two-step process, where an initial tessellation $\text{Del}(X)$, created for the vertices $X \in \mathcal{P}$, is refined until all constraining faces $f_c \in \mathcal{P}$ are recovered. This refinement is typically achieved through the recursive bisection of unrecovered constraints. Given a constraining face f_c additional Steiner points $\mathbf{x}_i$ are introduced on the surface of f_c. Given the introduction of a sufficiently large number of Steiner vertices, the constraining face is recovered as a union of faces in the underlying complex, such that $f_c = \bigcup_{f \in \text{Del}(X)} f$. See Figure 2.5 for an illustration of this process in $\mathbb{R}^2$. The use of Steiner triangulations also helps to resolve the issue of conformity for PLC's in higher-dimensions. Specifically, it is known that, in the worst-case, a Steiner triangulation of a general polyhedral domain $\mathcal{P}$ can be constructed through the addition of, at most, $O(n^2)$ Steiner vertices, where $n = |X|$ and X is the set of vertices in $\mathcal{P}$. Unfortunately, when additionally requiring that such tessellations are also Delaunay, these guarantees are substantially degraded. It is currently unknown whether a Steiner Delaunay triangulation can be constructed for a general polyhedral domain $\mathcal{P}$ in $\mathbb{R}^3$ using only $O(n^k)$ additional Steiner vertices, for some $k \geq 1$.

In practice, these upper bounds are discouraging, revealing the existence of pathological PLC's that require unacceptable levels of *over-refinement* when seeking to build conforming tessellations. The situation is somewhat ameliorated in many practical cases, where the required number of Steiner vertices is observed to be significantly less than the worst-case bounds [37, 38]. A more reasonable approach can be realised through the use of so-called *constrained Delaunay triangulations*, or CDT's, which relax the Delaunay criteria for simplexes adjacent to constraining faces. Specifically, Shewchuk [46] has shown that every *edge-protected* PLC in $\mathbb{R}^2$ and $\mathbb{R}^3$, that is, one in which all edges are strongly Delaunay, supports an associated CDT. Furthermore, Shewchuk [44, 45] has presented a number of efficient polynomial-time algorithms for the construction of such constrained triangulations in practice. I refer the reader to the expositions presented in [15] for further details.

Figure 2.6: Illustration of a restricted Delaunay triangulation for a curved domain in $\mathbb{R}^2$, showing, clockwise from top-left, (i) the bounding contour Σ and enclosed area Ω, (ii) the Delaunay triangulation $\mathrm{Del}(X)$ of a point-wise sampling of the contour $X \in \Sigma$, (iii) the associated Voronoi diagram $\mathrm{Vor}(X)$ highlighting the restricted Voronoi edges that intersect with Σ, and (iv) the restricted sub-complexes $\mathrm{Del}\,|_\Sigma(X)$ and $\mathrm{Del}\,|_\Omega(X)$ approximating the contour Σ and enclosed area Ω respectively.



2.2.2 Restricted Delaunay Tessellations

In practice, it is often desirable to tessellate and mesh general curved domains, consisting of collections of curves, surfaces and bounded volumes. To accommodate these structures, it is necessary to depart from the linear framework developed for PLC's in the preceding section and to introduce the notion of so-called *restricted* triangulations applicable to such curved structures. In this study, I restrict my attention to curved domains described by closed 2-manifolds Σ embedded in $\mathbb{R}^3$. Such geometries are common-place in the computational modelling and computer graphics communities.

Definition 2.20 (restricted Delaunay)**.** Let Σ be a closed 2-manifold embedded in $\mathbb{R}^3$, enclosing a bounded volume $\Omega \subset \mathbb{R}^3$. Let $\mathrm{Del}(X)$ be a full-dimensional Delaunay tessellation of a point-set $X \subseteq \Sigma$ and $\mathrm{Vor}(X)$ be its associated Voronoi diagram. The *restricted Delaunay surface triangulation* $\mathrm{Del}\,|_\Sigma(X)$ is a sub-complex of $\mathrm{Del}(X)$ including all 2-faces $f \in \mathrm{Del}(X)$ associated with an edge $\mathbf{v}_f \in \mathrm{Vor}(X)$ that intersects the surface, such that $\mathbf{v}_f \cap \Sigma \neq \emptyset$. The *restricted Delaunay volume triangulation* $\mathrm{Del}\,|_\Omega(X)$ is a sub-complex of $\mathrm{Del}(X)$ including all simplexes $\tau \in \mathrm{Del}(X)$ associated with an *internal* circumcentre $\mathbf{c} \in \Omega$.

First formally introduced by Edelsbrunner and Shah [22], restricted Delaunay tessellations are imbued with a number of desirable theoretical properties. It has been shown by a number of authors, including Cheng, Dey, Levine, Ramos and Ray [12, 13, 14] and

Boissonnat and Oudot [9, 10] that given a *sufficiently-dense* point-wise sampling[1] of the surface $X \subseteq \Sigma$ the restricted Delaunay tessellation $\text{Del}|_\Sigma(X)$ is guaranteed to be both geometrically and topologically representative of the underlying surface Σ. Specifically, it is known that, under such conditions, $\text{Del}|_\Sigma(X)$ is homeomorphic to the surface Σ, that the Hausdorff distance $\text{H}(\Sigma, \text{Del}|_\Sigma(X))$ is small and that $\text{Del}|_\Sigma(X)$ provides a good piecewise approximation of the geometrical properties of Σ, including its normals, area and curvature. Similar theoretical guarantees extend to the associated restricted volume triangulation $\text{Del}|_\Omega(X)$. The properties of restricted Delaunay tessellations are well documented in the literature, and I do not give a full account here. I refer the reader to the full expositions presented in [15, Chapters 12,13] for further details and proofs.

2.3 Algorithms for Delaunay Tessellations

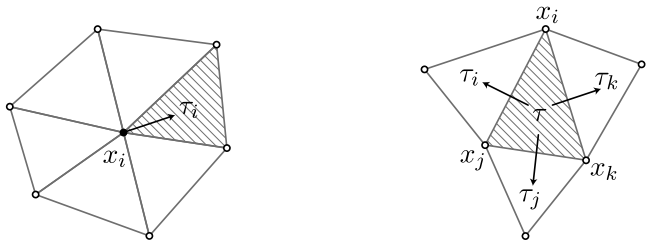
The usefulness of Delaunay tessellations in practice is tempered by the efficiency of the algorithms used for their construction and maintenance. A number of well known techniques have been developed, including the so-called *gift-wrapping*, *divide & conquer* and *incremental* approaches.

In the gift-wrapping approach, the Delaunay tessellation $\text{Del}(X)$ is constructed one element at a time, with a new element τ generated about an existing *base* face f by selecting a suitable apex vertex $\mathbf{x}_i \in X$. Suitable vertices are chosen to ensure that the new element τ satisfies the Delaunay property, and that the vertex is correctly oriented and visible from the base face f. Standard gift-wrapping is a relatively simple approach and can be made to run in $O(nt)$ time, where n is the number of vertices $|X|$ and t is the number of elements in the output $|\mathcal{T}|$. Note that for problems in $\mathbb{R}^3$ the size of the output can be quadratic, such that $|\mathcal{T}| = |X|^2$. Note also that gift-wrapping does not directly facilitate the *dynamic* maintenance of Delaunay tessellations, requiring that the full set of vertices X be available up-front. The gift-wrapping algorithm has been rediscovered independently by a number of authors, including, for example, early investigations by Frederick et al. [25] and McLain [34]. See also the extended discussions presented by Cheng, Dey and Shewchuk in [15].

Divide & conquer approaches for Delaunay tessellations are based on a recursive strategy, in which the set of vertices X is partitioned about a sequence of hyperplanes. Once the vertex subsets $Y_i \subset X$ are small enough, a local Delaunay tessellation $\text{Del}(Y_i)$ is constructed for each partition. The final tessellation $\text{Del}(X)$ is built by repeatedly merging the sub-tessellations across the splitting hyperplanes, ensuring that the Delaunay property is maintained in the merging region. Divide & conquer approaches are especially effective in $\mathbb{R}^2$, triangulating a set of points in $O(n \log(n))$ time, where $n = |X|$. Shewchuk [42] reports that divide & conquer methods outperform all alternative techniques, although this efficiency is not necessarily replicated in higher dimensions. Note also that divide & conquer techniques, like gift-wrapping, do not support dynamic tes-

[1] The point-wise sampling $X \subseteq \Sigma$ is required to be a so-called γ-*sample*, meaning that it is *sufficiently-dense* with respect to the local *medial-distance* induced by the enclosed volume Ω. These considerations are discussed in detail in Chapter 4.

Figure 2.7: Triangle-based data structures for Delaunay tessellations, showing (i) the vertex adjacency and (ii) triangle adjacency for a tessellation embedded in $\mathbb{R}^2$.



(i) (ii)

sellations, requiring the full set of vertices X be specified at the outset. A number of authors have contributed to the development of divide & conquer methods for Delaunay tessellations, including Lee and Schachter [32], Guibas and Stolfi [27], Dwyer [20] and Shewchuck [42].

Incremental techniques are based on a fundamentally different paradigm – seeking to maintain the Delaunay tessellation *dynamically*, through the incremental insertion, removal and update of vertices and elements. Incremental algorithms allow vertices to be inserted into or removed from a tessellation one at a time, applying a sequence of local transformations to ensure that the Delaunay property is preserved. In addition to their flexibility, incremental techniques can also achieve optimal algorithmic complexity, tessellating a set of points in the plane in $O(n \log(n))$ time, and a set of points in $\mathbb{R}^3$ in $O(n^2)$ time, where $n = |X|$. Incremental algorithms were first presented by Lawson in [31], who used a sequence of *edge-flip* transformations to restore the Delaunay property to a two-dimensional triangulation following an update. Bowyer [11] and Watson [47] are responsible for the so-called Bowyer-Watson approach that will form the body of the Delaunay tessellation framework developed in this study.

In this work, I pursue the development of incremental algorithms to build and maintain Delaunay tessellations in $\mathbb{R}^2$ and $\mathbb{R}^3$, since these methods best facilitate the type of adaptive construction, spatial search and dynamic updates required by meshing algorithms.

2.3.1 A Framework for Delaunay Tessellations

Adopting a so-called *triangle-based* data structure, the Delaunay tessellation $\mathrm{Del}(X)$ for a set of points $X \subset \mathbb{R}^d$ can be represented as a collection of vertices and d-simplexes. In this framework, the intermediate $(1, ..., d-1)$-simplexes are not stored explicitly, though they can be recovered as suitable subsets of the vertices in the d-simplexes. Adjacency information is maintained within the framework by explicitly tracking a limited subset of vertex and element connectivities. Each vertex $\mathbf{x} \in X$ stores a handle to a single vertex-adjacent d-simplex $\tau_j \cap \mathbf{x} \neq \emptyset$, and each d-simplex $\tau \in \mathrm{Del}(X)$ stores handles to both the vertices $\mathbf{x}_i \in \tau$, and the face-adjacent d-simplexes $\tau_j \cap \tau \neq \emptyset$. See Figure 2.7 for an illustration of this arrangement. The use of triangle-based data structures was first popularised by Shewchuk in [43] and is also used within the Computational Geometry

Algorithm 2.3.1 Vertex Insertion using the Bowyer-Watson Method

1: **function** BowyerWatson($\mathbf{x}_i$, Del(X))
2: Find a *conflicting* element $\tau_i \in \text{Del}(X)$, such that the open circumball of τ_i contains the point $\mathbf{x}_i$.
3: Start a breadth-first-search from τ_i, accumulating the set of elements in the *conflict cavity* $\mathcal{C} \subseteq \text{Del}(X)$, such that the vertex $\mathbf{x}_i$ lies within the open circumballs of all $\tau_j \in \mathcal{C}$.
4: Delete all elements $\tau_j \in \mathcal{C}$.
5: Re-triangulate the cavity $\mathcal{C}$ about the vertex $\mathbf{x}_i$. A new element τ_k is created for all *boundary* faces $f \in \mathcal{C}$.
6: **end function**

Algorithms Library (`CGAL`) [8]. Note also that the triangle-based data structure for Del(X) described here can be interpreted as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where each d-simplex $\tau \in \text{Del}(X)$ is a graph-vertex $\mathbf{v}_i \in \mathcal{V}$ and each pair of face-adjacent d-simplexes in Del(X) is a graph-edge $e_i \in \mathcal{E}$. This graph-based interpretation of Del(X) allows the standard repertoire of algorithms and traversal techniques developed in graph theory to be used directly in the discussions of the triangulation algorithms that follow.

The triangle-based data structure for Del(X) is designed to facilitate efficient implementations for the various spatial and topological queries that are required in later chapters. Specifically, note that given a d-simplex $\tau \in \text{Del}(X)$ the proposed framework supports $O(1)$ access to both the vertices of the element and its face-adjacent neighbours. Given a vertex $\mathbf{x} \in X$, access to an adjacent d-simplex in Del(X) is also available in $O(1)$ time. These operations allow for the creation of a number of schemes to efficiently *walk* the triangulation from a given seed. For example, given a seed element $\tau \in \text{Del}(X)$ the local-neighbourhood of τ can be traversed using standard breadth-first or depth-first search in $O(m)$ time and space, where m is the number of elements visited. Given a seed vertex $\mathbf{x} \in X$ it is possible to use both vertex- and element-adjacency to traverse the *ring* of m elements adjacent to $\mathbf{x}$ in $O(m)$ time and space. These, and other similar traversals, will be used extensively in the development of the triangulation and meshing algorithms presented in this study.

2.3.2 The Bowyer-Watson Algorithm

The Bowyer-Watson algorithm is an incremental method for the insertion of a new vertex into an existing Delaunay tessellation Del(X). Given a new vertex $\mathbf{x}_i$, inserted into the underlying point-set, such that $X' = X \cup \{\mathbf{x}_i\}$, the Bowyer-Watson algorithm constructs an updated Delaunay tessellation Del(X') based on a sequence of *local* transformations applied to the existing tessellation Del(X). The algorithm is summarised in Algorithm 2.3.1 and a simple example of its operation in $\mathbb{R}^2$ is shown in Figure 2.8.

The algorithm first searches the existing tessellation Del(X) for the d-simplex τ_i that encloses the new vertex $\mathbf{x}_i$. In this study, I simplify this process by ensuring that the tessellation includes a large bounding simplex enclosing the full set of points to be tessellated. Clearly, as a result, $\mathbf{x}_i \in \text{Conv}(X)$ and the enclosing simplex $\tau_i \in \text{Del}(X)$ always exists. Efficient traversal methods to locate this enclosing simplex are discussed in further detail in subsequent sections. The aim of the insertion process is to generate a new

Figure 2.8: Vertex insertion via the Bowyer-Watson approach, showing, clockwise from top-left: (i) location of the enclosing triangle for the new vertex $\mathbf{x}_i$, (ii) identification of the *conflict* cavity $\mathcal{C} \subseteq \mathrm{Del}(X)$, (iii) re-triangulation of the cavity $\mathcal{C}$ about the new vertex $\mathbf{x}_i$, and (iv) updated triangulation $\mathrm{Del}(X')$.

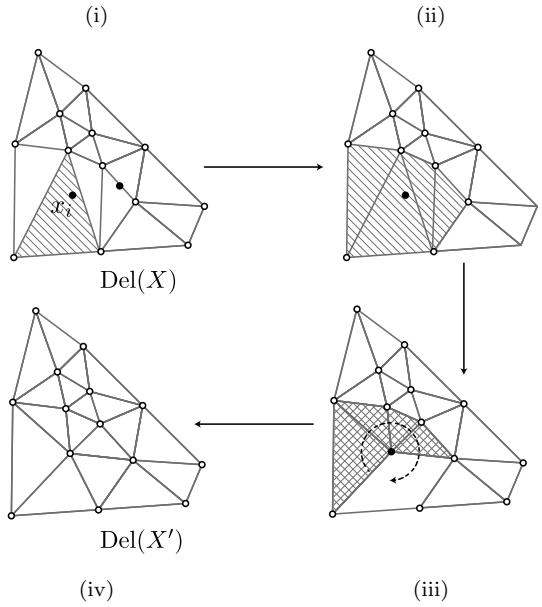
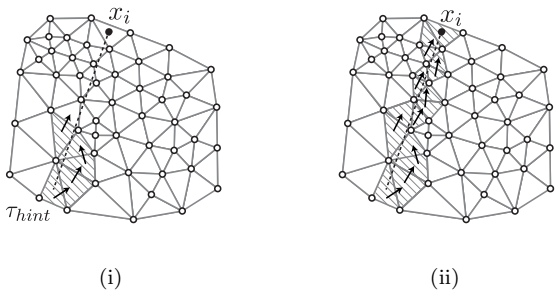


Figure 2.9: Point location via the directed traversal method, showing (i) initiation of the traversal from the initial guess τ_{hint}, and (ii) full traversal along the line $\mathcal{L}$ adjoining the element τ_{hint} and the point $\mathbf{x}_i$.



tessellation $\mathrm{Del}(X')$ in which all simplexes are Delaunay. The Bowyer-Watson method proceeds by first identifying and removing any exiting elements $\tau_j \in \mathrm{Del}(X)$ that, due to the position of the new vertex $\mathbf{x}_i$, violate the Delaunay property. This *conflict cavity* $\mathcal{C} \in \mathrm{Del}(X)$ can be found efficiently by starting a breadth-first-search about the enclosing simplex τ_i and accumulating all simplexes $\tau_j \in \mathrm{Del}(X)$ that include the vertex $\mathbf{x}_i$ within their open circumballs. The cavity $\mathcal{C}$ is known to be *star-shaped*, meaning that the new vertex $\mathbf{x}_i$ is visible from all bounding faces $f \in \mathcal{C}$. This result suggests a simple method for the re-triangulation of the cavity $\mathcal{C}$, in which a new simplex τ_k is created between each boundary facet $f \in \mathcal{C}$ and the vertex $\mathbf{x}_i$. It can be shown that the resulting tessellation $\mathrm{Del}(X') = \mathrm{Del}(X \cup \{\mathbf{x}_i\})$ is Delaunay. I refer the reader to [15, Chapter 2–5] for a full proof of correctness. Using the triangle-based data structure discussed previously, the Bowyer-Watson algorithm can be implemented to run in $O(|\mathcal{C}|) + P$ time, where P is the

Algorithm 2.3.2 Point Location using Directed Traversal

```
1: function DirectedTraversal(x_i, Del(X), τ_i)
2:     Form line ℒ from the centre of τ_i to x_i.
3:     while (τ_i ⋂ x_i = ∅) do
4:         for all (f ∈ τ_i) do
5:             if (f ⋂ ℒ ≠ ∅) then                ▷ {'walk' over intersecting face}
6:                 Set τ_i to face-adjacent neighbour τ_j; break
7:             end if
8:         end for
9:     end while
10: end function
```

cost of locating the enclosing simplex. Excluding the cost of point location, which will be dealt with subsequently, the remaining $O(|\mathcal{C}|)$ cost is optimal, considering that the updated tessellation $\text{Del}(X')$ requires the formation of $O(|\mathcal{C}|)$ new simplexes.

2.3.3 Directed Traversal and Point Location

Given a point $\mathbf{x}_i \in \text{Conv}(X)$ the location of the enclosing simplex $\tau_i \in \text{Del}(X)$ is an important operation required by a variety of triangulation and meshing algorithms, including the Bowyer-Watson method for point insertion presented above. The use of a naïve approach, in which all simplexes $\tau_i \in \text{Del}(X)$ are tested, leads to unacceptably poor algorithmic complexity, requiring, for example, $O(n)$ time in $\mathbb{R}^2$ and $O\big(n^2\big)$ time in $\mathbb{R}^3$, where $n = |X|$. Much faster performance can typically be achieved in practice through the use of a so-called *directed traversal* of $\text{Del}(X)$. Given a target point $\mathbf{x}_i$ and a *hint* simplex τ_{hint}, a directed traversal *walks* through the tessellation $\text{Del}(X)$, only visiting simplexes $\tau \in \text{Del}(X)$ that overlap a path $\mathcal{L}$, drawn between the starting simplex τ_{hint} and the target point $\mathbf{x}_i$. It is straightforward to develop an algorithm that only checks simplexes that intersect with the linear search path by simply *walking* from one simplex to its next intersecting facet-adjacent neighbour. Such an algorithm, implemented using the triangle-based data structure described previously, takes $O(m)$ time per point location query, where m is the number of simplexes visited during the traversal. Clearly, if the hint is chosen sufficiently close to the target point $\mathbf{x}_i$, optimal $O(1)$ complexity is achieved. Many triangulation and meshing algorithms can be structured to facilitate the proximity of these hints in practice, leading to optimal performance. The directed traversal method for point location is summarised in Algorithm 2.3.2, and a simple example illustrating its operation is presented in Figure 2.9. Walking methods for point location have been investigated by a number of authors, including, for example, Devilliers, Pion and Teillaud [18, 19].

2.3.4 Efficient Bulk Loading

While the Bowyer-Watson and Directed Traversal algorithms discussed thus far provide a framework for the incremental assemblage of Delaunay tessellations, it is also commonly required that a full tessellation $\text{Del}(X)$ be built given a known set of points X. Clearly, such a tessellation can be constructed incrementally, simply by inserting each vertex $\mathbf{x}_i \in X$ into $\text{Del}(X)$ one-by-one, but a careful analysis of the expected runtime shows

that this simple method may lead to unacceptably slow performance in some cases. When a new vertex is inserted using the Bowyer-Watson algorithm, a conflict cavity $\mathcal{C} \subseteq \mathrm{Del}(X)$ must be re-triangulated. It is known that pathological configurations exist in which the insertion of a single vertex $\mathbf{x}_i$ into an existing tessellation $\mathrm{Del}(X)$ can result in large cavities, such that $|\mathcal{C}| = O(|\,\mathrm{Del}(X)|)$. In the worst case, the insertion of a vertex $\mathbf{x}_i$ can lead to the creation of $O(n)$ new simplexes, where $n = |X|$. Clearly, a sequence of n such insertions leads to $O\big(n^2\big)$ overall worst-case performance.

The expected performance of incremental construction can be improved through the use of *randomisation*. Given a set of vertices X, the Delaunay tessellation is constructed by applying a randomised permutation $\mathcal{Q}$ to the list of vertices, such that the vertex $\mathbf{x}_{\mathcal{Q}_i}$ is added to $\mathrm{Del}(X)$ at the i-th step of the assembly. Use of such a scheme reduces the probability of encountering pathological configurations in practice and improves the expected running times considerably. Specifically, for triangulations in the plane, it has been shown by Clarkson and Shor [16] that the expected performance of the incremental construction algorithm is improved to $O(n) + P$, where P is the cost of point location. For point-sets in $\mathbb{R}^3$, the guarantees, unfortunately, are not as strong, with some pathological point-sets X known to produce tessellations that are quadratically large, such that $|\,\mathrm{Del}(X)| = O\big(n^2\big)$. Due to the size of the output in these cases, it is clearly not possible to improve on the $O\big(n^2\big)$ runtime, using any method. In practice, randomisation is still known to help reduce the occurrence of such pathological configurations, unless they are unavoidable, and it is reported by Dwyer [21] and Amenta, Choi and Rote [2] that optimal $O(n) + P$ performance is often observed for a wide class of tessellations in $\mathbb{R}^3$ in practice. The practical performance and theoretical complexity of incremental schemes for vertex insertion have been a topic of extensive research in the literature, and I refer the reader to the detailed expositions presented in [15, Chapter 3,5] for additional information.

In addition to the performance of vertex insertion, the cost of point location is an equally important contributor to the overall expense of incremental construction methods. In [16], Clarkson and Shor present an optimal structure for point location known as a *conflict-graph*, and demonstrate that the use of such an device leads to optimal $O(n \log(n))$ expected performance for the construction of Delaunay tessellations in the plane. In this study I pursue a somewhat simpler and practically efficient scheme, based on the use of the so-called *Biased Randomised Insertion Orders*, or BRIO's, introduced by Amenta et al. in [2]. Recalling that the cost of traversal-based point location is $O(m)$, where m is the number of visited simplexes, the expense of successive point location queries can be mitigated by ensuring that a suitable *starting hint* is available for each subsequent traversal. This implies that the vertices X should be added to $\mathrm{Del}(X)$ according to a schedule that furnishes their *spatial-locality*, such that successive vertices are relatively *close* to one another in $\mathbb{R}^d$. Considering such an ordering, optimal $O(1)$ expected time point location can be achieved for the insertion of two successive vertices $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, simply by ensuring that a simplex adjacent to the preceding vertex $\mathbf{x}_i$ is used as the starting hint when traversing to find the enclosing simplex for the new vertex $\mathbf{x}_{i+1}$.

In the BRIO framework presented in [2], Amenta et al. balance the objectives of purely random and spatially local insertion orders by modifying an ordering designed for spatial locality to incorporate a small random bias. Given a set of points X to tessellate, the points are partitioned into a sequence of *rounds*, where the vertices in each round are added to the tessellation successively. The rounds are constructed from the bottom-up, such that all vertices in X are assigned to the final round with probability $1/2$, with the remaining vertices then assigned to the second-last round with probability $1/2$, and so on. In total, $\log_2(|X|)$ rounds are needed. The structure of the rounds themselves provide sufficient randomisation to preclude the occurrence of worst-case behaviour, therefore allowing an efficient spatially-local vertex ordering to be used *within* each round. In this thesis, I adopt a spatial ordering based on adaptive Hilbert curves [29], where the vertices within each round $X_i \subseteq X$ are first partitioned between the leaves of an associated 2^d-tree $\mathcal{S}$, and then subsequently inserted into $\text{Del}(X)$ through a traversal of $\mathcal{S}$ in Hilbert order. 2^d-trees are considered in detail in Section 2.4. A full review of the incremental framework for Delaunay tessellations, including discussions of conflict graphs and BRIO's, is presented in [15, Chapter 5].

2.3.5 Implementation & Discussions

A new triangulation framework – `TRIPOD` – supporting the incremental construction and maintenance of Delaunay tessellations embedded in $\mathbb{R}^2$ and $\mathbb{R}^3$ has been implemented. Support for dynamic vertex insertion, point location and bulk construction is achieved through use of the Bowyer-Watson, directed-traversal and BRIO-based bulk-loading techniques outlined earlier in this chapter. The accuracy and performance of the new `TRIPOD` framework is assessed through a series of comparative tests, where the Delaunay triangulation routines provided in the well-known `CGAL` and `QHULL` packages, due to Boissonnat, Devillers, Pion, Teillaud, and Yvinec [8] and Barber, Dobkin and Huhdanpaa [4] respectively, are used to provide benchmark results. The `CGAL` package provides support for Delaunay tessellations in $\mathbb{R}^2$ and $\mathbb{R}^3$ directly, based on an incremental paradigm similar to that used in the present work. In contrast, the `QHULL` package – an implementation of the Quickhull algorithm for convex hulls – facilitates the construction of Delaunay tessellations in $\mathbb{R}^d$ *indirectly*, through a projection of the downward-facing facets of the 'lifted' hull from $\mathbb{R}^{d+1}$ onto $\mathbb{R}^d$, as per the discussions presented in Section 2.2.

I have made use of a *bounding-simplex* approach in the new implementation, in which a large initial simplex is created to enclose all vertices in the point-set $X \subset \mathbb{R}^d$ to be triangulated. This is achieved through the addition of an extra $d+1$ vertices X_{init}, positioned such that their resulting tessellation $\text{Del}(X_{\text{init}})$ is a single simplex that fully encloses all points in X. Such a procedure simplifies the strategy used for vertex insertion, guaranteeing that all subsequent vertices are inserted into the 'interior' of $\text{Del}(X \cup X_{\text{init}})$, alleviating the need to incrementally manage the boundary of the tessellation. `CGAL`, on the other hand, explicitly maintains the associated hull, $\text{Conv}(X)$, along with its triangulation $\text{Del}(X)$, and, as a result, supports fully dynamic tessellations in arbitrary domains. `QHULL`, as a convex hull code, also returns $\text{Del}(X)$ as a direct tessellation

Figure 2.10: Comparative Delaunay tessellation results in $\mathbb{R}^2$, showing the NEWZEALAND, INDIA, AUSTRALIA, GREENLAND, COASTLINE and ISLANDS test cases. Note that the *trimmed* tessellations are shown, with elements adjacent to the bounding simplex $\mathcal{T}_{\text{init}} = \text{Del}(X_{\text{init}})$ removed.

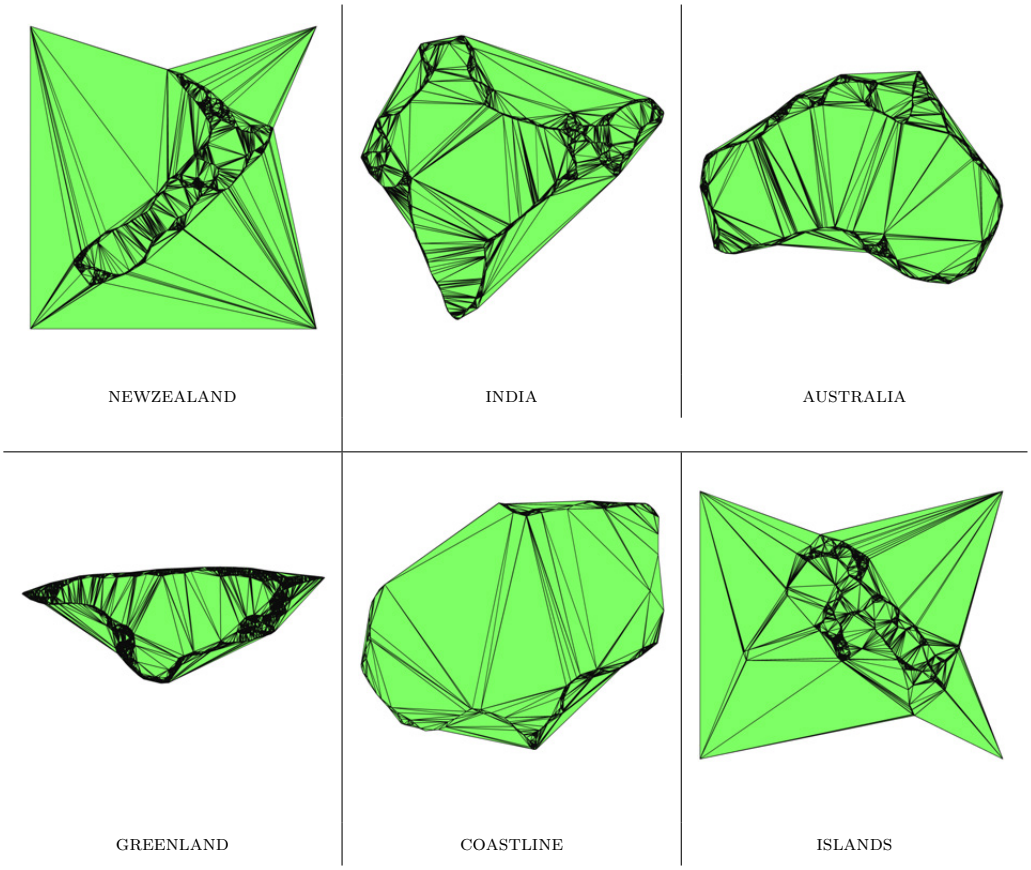


Table 2.1: Timing results associated with Figure 2.10, demonstrating the performance of the methods developed in the present study, and the existing `CGAL` and `QHULL` packages.

Test Problem	$\lvert X\rvert$	$\lvert\text{Del}(X)\rvert$	Runtime (s)		
			`TRIPOD`	`CGAL`	`QHULL`
NEWZEALAND	11,668	23,331	0.034	**0.032**	0.071
ISLANDS	7,074	14,143	0.020	**0.018**	0.038
AUSTRALIA	34,417	68,829	**0.099**	0.100	0.187
GREENLAND	39,148	78,291	**0.108**	0.116	0.215
COAST	11,002	21,999	0.035	**0.031**	0.061
INDIA	14,512	29,019	0.044	**0.040**	0.066

Figure 2.11: Comparative Delaunay tessellation results in $\mathbb{R}^3$, showing the VENUS, DINOSAUR, HAND, WOODTHINKER, BLADE and BIMBA test cases. Note that the underlying surface models are shown, rather than $\mathrm{Del}(X)$, as direct visualisations of $\mathrm{Del}(X)$ in $\mathbb{R}^3$ are difficult to interpret.

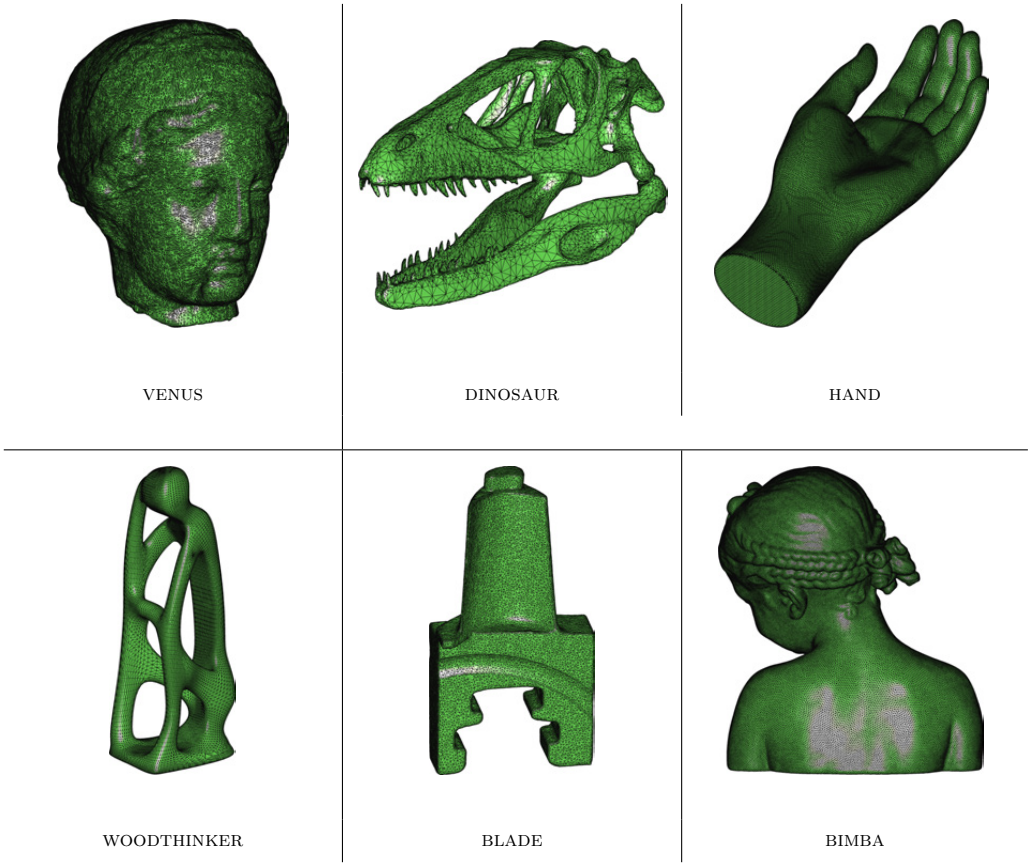


Table 2.2: Timing results associated with Figure 2.11, demonstrating the performance of the methods developed in the present study, and the existing `CGAL` and `QHULL` packages.

Test Problem	$\lvert X\rvert$	$\lvert\mathrm{Del}(X)\rvert$	Runtime (s)		
			`TRIPOD`	`CGAL`	`QHULL`
VENUS	67,184	456,883	1.000	**0.872**	3.040
BLADE	25,002	170,386	0.361	**0.326**	0.688
BIMBA	74,768	486,820	1.080	**1.000**	2.220
DINOSAUR	27,731	189,134	0.400	**0.362**	0.773
HAND	38,222	266,489	0.621	**0.544**	2.140
WOODTHINKER	14,010	98,972	0.199	**0.182**	0.388

of Conv(X), though it does not support incremental modification. The algorithms in TRIPOD have been designed to target mesh generation applications specifically, where the full tessellation Del($X \cup X_{\text{init}}$) is always subsequently *trimmed* to conform to a particular geometry through the removal of a set of 'external' simplexes. As such, there is currently no support (or need) in TRIPOD for tessellations that conform to Conv(X) explicitly.

The use of finite numerical precision is known to lead to serious issues of robustness and reliability when implementing Delaunay-based algorithms, and such matters are addressed in the present work through the use of adaptive floating point arithmetic, robust geometrical predicates and direct perturbation for degenerate co-spherical vertices. Robust implementations of the necessary 'orientation' and 'in-circumball' queries, required by the point-location and Boywer-Watson sub-routines, are realised through use of the high-quality adaptive-precision techniques presented by Shewchuk [41]. As detailed in Shewchuk's original paper, these adaptive routines offer an excellent compromise between accuracy and efficiency, performing computations using efficient double-precision arithmetic when results are unambiguous, while smoothly transitioning to an expensive variable-precision computation when required to preserve accuracy. Shewchuk offers an open-source implementation of these routines, and his procedures are used in the present work without alteration.

2.3.6 Experimental Comparisons

The effectiveness of the new TRIPOD framework was assessed through a series of comparative experimental studies, contrasting the performance and output of the new algorithms with that of the CGAL and QHULL packages. Overall, a set of twelve test cases were examined, based on point-sets embedded in $\mathbb{R}^2$ and $\mathbb{R}^3$. In each case, the Delaunay tessellation was constructed in bulk for a given set of vertices $X \subset \mathbb{R}^d$, with the runtime measured in each case. Additionally, following the approach introduced by Boissonnat et al. [8], a number of geometrical and topological tests were conducted, ensuring, firstly, that the open-circumball of each simplex $\tau \in \text{Del}(X)$ was empty, and secondly that each internal $(d-1)$-facet was shared by exactly two simplexes $\tau_i, \tau_j \in \text{Del}(X)$. Note that these tests were implemented using a simple exhaustive search, providing a verification of the triangle-based data-structures used in the new implementations. In all cases, it was found that the algorithms in TRIPOD constructed the Delaunay tessellations correctly, passing all geometric and topology-based tests.

Results for the two-dimensional test problems are presented in Figure 2.10, showing both the output generated using TRIPOD in addition to a tabulation of the associated performance metrics for the three algorithms tested. These test cases are sourced from number of complex geo-spatial datasets, in which the point-wise representation of the coastlines of various geographical features are triangulated. Note that the triangulations shown are *trimmed*, in which elements adjacent to vertices in the bounding simplex X_{init} have been removed. The resulting trimmed triangulations are not necessarily convex as a result. An analysis of the runtime performance of the various algorithms demonstrates that the algorithms in TRIPOD are efficient for problems in $\mathbb{R}^2$, being roughly equivalent

to the high-quality implementation provided in the CGAL package. The QHULL library, in contrast, is found to be consistently slower than either of the incremental schemes, and is typically outperformed by a factor of approximately 2. This slow-down is not unexpected, considering the indirect nature of the triangulation algorithm implemented in QHULL.

Results for the three-dimensional test problems are presented in Figure 2.11, showing both the underlying surface models to be tessellated and a tabulation of the associated performance metrics for the three algorithms tested. These test problems are based on a series of triangulated surface geometries, sourced from various application areas including computational simulation and computer graphics. An analysis of the runtime performance of the various algorithms demonstrates that the algorithms in TRIPOD are reasonably efficient for problems in $\mathbb{R}^3$, being approximately 5–15% slower than the high-quality implementations provided in the CGAL package. The specific cause of this slow-down is unclear, and is a topic for future investigations. Consistent with the results in $\mathbb{R}^2$, the QHULL package is seen to be notably slower across the full set of test problems, being outperformed by the incremental algorithms by a factor of 2–3.

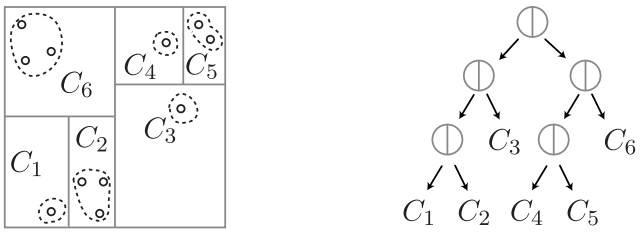
Overall, these results confirm the effectiveness of the new TRIPOD library, showing that the new implementation is capable of efficiently and robustly triangulating the type of large-scale two- and three-dimensional datasets associated with meshing problems.

2.4 Spatial Subdivisions & Trees

Geometrical structures based on recursive spatial subdivisions are useful complements to the Delaunay-based tessellations presented earlier in this chapter. Rather than seeking to tessellate point-sets, these so-called *spatial-trees* are designed to store generalised collections of geometrical entities, including, for example, points, lines, simplexes, and even more complex compound entities such as polyhedrons and polyhedral complexes. Spatial trees are inherently hierarchical structures, organised as ordered sets of bounding hyper-rectangles, referred to simply as bounding rectangles in the discussions that follow. Each bounding rectangle in the tree encloses a subset of the geometrical entities to be represented. At its top-most level, a spatial tree consists of a *root* node – a large bounding rectangle that encloses the full set of geometric entities in the collection to be stored. The remainder of the tree is comprised of a hierarchy of *nodes*, each enclosing a subset of the geometrical entities in the collection. Each node in the tree is associated with both *parent* and *child* nodes, referring to larger and smaller bounding rectangles that immediately precede, or succeed a given node in the hierarchy. Nodes at the bottom-most level of the tree are those without children, and are commonly referred to as *leaf* nodes. See Figure 2.12 for an example of a simple spatial tree in $\mathbb{R}^2$.

Due to their hierarchical nature, spatial trees facilitate efficient spatial search queries, including, for example, finding the set of entities that lie within a given halo region or performing various intersection tests between entities in the collection. Further elaboration of the specific spatial queries used throughout this study will be presented in subsequent sections. In broad terms, spatial trees achieve their efficiency through *spa-*

Figure 2.12: Illustration of a spatial tree $\mathcal{S}$ for a simple point-wise collection C, showing (i) the recursive rectangular subdivision of the bounding rectangle, and (ii) the associated binary tree hierarchy.



(i) (ii)

tial localisation – a segregationist process in which entities are partitioned according to their relative distance to others in the collection. This localisation allows spatial queries to be executed visiting only a local subset of the full geometrical collection, leading to significant gains in efficiency. Spatial trees are used extensively throughout this thesis to improve the performance of algorithms based on spatial queries and traversals.

2.4.1 Quadtrees, Octrees & 2^d-trees

The simplest type of spatial trees are based on a multi-dimensional bisection strategy. Let C be a collection of geometrical entities. Consider a spatial tree $\mathcal{S}$, designed to store the collection C, where $\mathcal{S}$ consists of a hierarchy of rectangular nodes R^k. Given an axis-aligned parent rectangle $R^k \in \mathcal{S}$, enclosing a subset $C^k \subseteq C$ of the full collection, a set of 2^d child sub-rectangles R_i^{k+1} are created by simply bisecting R^k about its midpoint using a series of splitting planes orthogonal to each axis direction. Restricting attention, firstly, to the storage of point-wise entities only, each child rectangle R_i^{k+1} encloses a sub-collection $C_i^{k+1} \subseteq C^k$, such that all entities in the set C_i^{k+1} are fully enclosed by the sub-rectangle R_i^{k+1}. Trees based on multi-dimensional bisection are well-known, with the classical *quadtree* and *octree* structures being specific examples in $\mathbb{R}^2$ and $\mathbb{R}^3$. In general, these types of spatial trees are referred to as 2^d-trees, referencing the number of child nodes spawned from each parent in the tree. It is important to note that 2^d-trees represent a so-called *complete* decomposition of a bounding space, in which the space enclosed by each parent node in the tree is recovered as the union of the spaces spanned by its children, such that $R^k = \bigcup_{i=1}^{2^d} R_i^{k+1}$.

The 2^d-trees used in this study are built using a simple top-down approach, in which a full collection of entities C is inserted into a sufficiently large bounding rectangle R^0, which constitutes the root of the tree. The tree is grown recursively, with a new set of child rectangles created for any rectangle R^k that it too populous, such that $|C^k| > \gamma$, where $\gamma \in \mathbb{Z}^+$ is a positive constant dictating the maximum allowable number of entities enclosed by a single node in the tree. The spatial trees introduced in this study are variants in which entities are only stored in the leaf nodes. As a consequence, when a parent node R^k is split, its associated sub-collection C^k is directly distributed between the

Algorithm 2.4.1 Spatial Tree Construction

```
1: function MakeTree(C, γ, S)
2:     S, Q_S ← R^0, C.                          ▷ {initialise tree, refinement queue}
3:     while ((k ← Q_S) ≠ ∅) do
4:         if (|C^k| > γ) then                    ▷ {refine node if too populous}
5:             Call RefineNode(R^k, C^k, R^{k+1}, C^{k+1})
6:             Update tree S ← R_i^{k+1}.
7:             Update queue Q_S ← R_i^{k+1}.
8:         end if
9:     end while
10: end function

1: function RefineNode(R^k, C^k, R^{k+1}, C^{k+1})          ▷ {node refinement}
2:     Split R^k into child nodes R_i^{k+1}.
3:     Split C^k into child sub-collections R_i^{k+1} ← C_i^{k+1}.
4: end function
```

new sub-collections C_i^{k+1} associated with its children. The final tree therefore consists of a hierarchy of *empty* non-leaf nodes, with the full collection of entities distributed amongst a set of sub-collections $C_i^l \subseteq C$ associated with the leaves of the tree. These final sub-collections are small, such that $|C_i^l| \leq \gamma$. A general procedure for tree construction is summarised in Algorithm 2.4.1.

At each *level* in the tree, a maximum of $O(|C|)$ work is done repartitioning the collection C amongst the new child nodes. This implies that the full tree can be built with a worst-case complexity of $O(nh)$, where $n = |C|$ and h is the *height* of the tree. For collections that are *well-distributed* it can be shown [23, 24] that the *expected* tree height is $O(\log(n))$, leading to an efficient $O(n \log(n))$ algorithm for tree construction. It should be noted that, for 2^d-trees, the tree height h is *not* bound with respect to $|C|$ in the worst-case, and, as such, despite the excellent expected performance described above, it is possible to construct pathological collections for 2^d-trees that lead to extremely poor performance. While these pathological inputs are exceedingly rare in practice, mildly sub-optimal performance is often observed, through the creation of *empty* sub-trees, in which a region of the tree, including its associated leaf nodes, are bereft of entities from the collection C. This wasteful over-refinement is a consequence of the simple midpoint splitting strategy associated with 2^d-trees and will be discussed in further detail in subsequent sections.

2.4.2 Kd-trees and Variations

More sophisticated types of spatial trees can be built through consideration of the *spatial-distribution* of the collection of entities to be stored. The well-known KD-tree, introduced by Bentley [7], pursues this argument, utilising a *median-based* bisection strategy to split nodes in the hierarchy. Such an approach ensures that the sub-collections associated with each node in the tree are partitioned equally, regardless of their spatial distributions. Specifically, given an axis-aligned parent rectangle $R^k \in \mathcal{S}$, enclosing a subset $C^k \subseteq C$ of the full collection C, a *pair* of sub-rectangles R_1^{k+1} and R_2^{k+1} are created by bisecting the rectangle R^k about a splitting plane orthogonal to the j-th median of C^k, where j is a coordinate direction, chosen in a cyclic fashion to ensure that subsequent

levels in the tree are split about different axes. Following the same process outlined previously for the 2^d-tree, the sub-collection C_i^{k+1} associated with the i-th child of R^k consists of entities fully enclosed by the sub-rectangle R_i^{k+1}. Conventional KD-trees can be built following an adaptation of the top-down procedure previously outlined for 2^d-trees, where an additional median finding step is performed each time a node is split. The resulting tree is guaranteed to be *well-balanced* and will not, as a result, contain more than $O(n)$ nodes. In addition to achieving an optimal expected tree height, $h = O(\log(n))$, consistent with the 2^d-tree described previously, the KD-tree ensures that the tree height remains bounded in the worst-case.

In this work I introduce a KD-tree variant differing from the conventional structure of Bentley in terms of the way that node splits are treated. In the new tree, given an axis-aligned parent rectangle $R^k \in \mathcal{S}$, enclosing a subset C^k of the full collection C, a pair of sub-rectangles R_1^{k+1} and R_2^{k+1} are created by bisecting R^k about a *pivot* plane positioned along the longest dimension of the rectangle R^k, orthogonal to the axis. Following the *sliding-midpoint* strategy of Mount, Arya and Maneewongvatana [33, 35] the pivot is positioned according to a two-stage process. The pivot is first placed at the midpoint of the longest dimension of R^k, and is accepted in this position provided that the splitting plane intersects the sub-collection C^k, such that both $C_1^{k+1} \neq \emptyset$ and $C_2^{k+1} \neq \emptyset$. If an empty child sub-collection is found, the pivot is repositioned by *sliding* the splitting plane towards the closest entity in C^k, until a strict intersection is achieved. Compared to the median split of the conventional KD-tree, the sliding-midpoint strategy is cheaper to evaluate, and typically improves the aspect ratio of the resulting sub-rectangles. Furthermore, since the sub-collections associated with the new child nodes are guaranteed to be non-empty, the worst-case performance of the KD-tree is preserved, ensuring that the resulting tree contains $O(n)$ nodes and that the maximum height is bounded.

Additionally, I employ a new *node-shrinking* strategy, inspired by the R-tree framework of Guttman [28], in which the set of bounding sub-rectangles associated with the tree are *shrunk* such that they constitute only a *minimal* covering of the entities in their associated sub-collections. Specifically, when a parent rectangle $R^k \in \mathcal{S}$ is split, according to the sliding midpoint method described previously, two new sub-collections C_1^{k+1} and C_2^{k+1} are induced about the splitting plane. Rather than simply constructing the new sub-rectangles R_1^{k+1} and R_2^{k+1} based on a full bisection of the rectangle R^k about the splitting plane, as per conventional KD-tree strategies, I instead form R_1^{k+1} and R_2^{k+1} directly, as minimal axis-aligned bounding rectangles for the new sub-collections C_1^{k+1} and C_2^{k+1}. Note that, following such a strategy, the new rectangles R_1^{k+1} and R_2^{k+1} are typically smaller than their associated parent rectangle R^k along *all* axes. Importantly, such a splitting strategy introduces *void* regions into the tree, in which space bounded by a parent rectangle is excluded from any of the sub-trees associated with its children. The use of this node-shrinking strategy often helps to further enhance spatial locality in the tree, and, as will be discussed in subsequent sections, can improve the performance of spatial queries by reducing the number of sub-rectangles that overlap with a given search window. See Figure 2.13 for an illustration of this node shrinking strategy for a

Figure 2.13: Example of the proposed *node-shrinking* technique showing (i) an un-split parent rectangle R^k and associated collection C^k, (ii) new child sub-collections C_1^{k+1} and C_2^{k+1} induced about the splitting plane, and (iii) new child sub-rectangles R_1^{k+1} and R_2^{k+1}, formed as minimal bounding rectangles for the new sub-collections C_1^{k+1} and C_2^{k+1}.

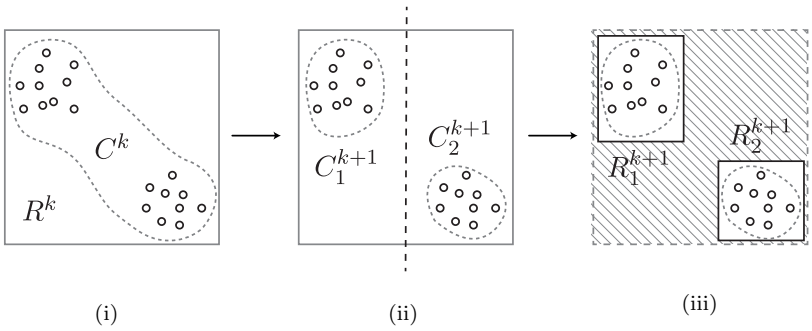
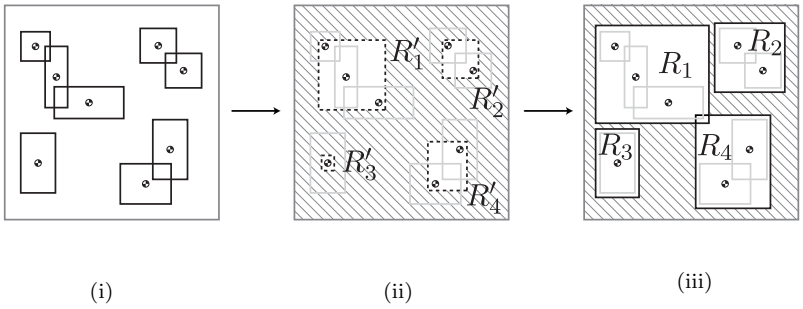


Figure 2.14: Illustration of the proposed *node-inflation* method for collections of AABB's, showing (i) a set of AABB's in a general collection C, (ii) an intermediate *point-wise* spatial tree, consisting of 4 nodes $\{R'_1, ..., R'_4\}$, formed using the centroids of the AABB's in C, and (iii) the final AABB-tree, in which the nodes $\{R_1, ..., R_4\}$ have been inflated to fully enclose all AABB's in their respective sub-collections.



simple node split in $\mathbb{R}^2$. The node shrinking strategies investigated in this work appear to be related to the so-called *box-decomposition* trees of Ayra et al., introduced in [3]. Consistent with the strategy presented previously for the 2^d-tree, the KD-trees developed in this study store entities in their leaf nodes only, and are refined to satisfy a maximum node population threshold, such that $|C_i^l| \leq \gamma$ for all leaf nodes $l \in \mathcal{S}$. Construction of KD-trees follows the general procedure summarised in Algorithm 2.4.1.

2.4.3 Storage of General Entities

The 2^d- and KD-tree types presented previously can be extended to handle collections of generalised geometrical entities, rather than the simple collections of point-wise data addressed thus far. Noting that any general geometrical entity can be represented in terms of a minimal axis-aligned bounding 'box', I extend the 2^d- and KD-tree types to store collections of such boxes. Following the typical parlance, I refer to these trees as Axis-Aligned-Bounding-Box trees, or, more frequently, AABB-trees. Note that the AABB's stored in the tree are associated specifically with entities in the collection C, and

should not be confused with the sub-rectangles $R \in \mathcal{S}$ that constitute the nodes of the spatial tree itself. Firstly, given a spatial tree $\mathcal{S}$, recall that each sub-rectangle $R^k \in \mathcal{S}$ is required to *fully enclose* all entities in its associated sub-collection C^k. Considering the case where C is a collection of AABB's, note that each rectangle R^k is required to fully enclose the AABB's for all entities $\mathbf{c}_i \in C^k$, rather than simply enclosing a collection of point-wise coordinates as has been discussed in previous sections. Importantly, this condition implies that adjacent nodes in the tree may *overlap* in certain cases, occurring if, for example, a collection of AABB's are encountered that *straddle* a splitting plane. In general, it is at times impossible to reposition the splitting planes to guarantee that a non-overlapping subdivision is induced, requiring that a generalised type of spatial tree incorporating such node-to-node overlap be considered. Such configurations are typical of the R-tree framework introduced by Guttman in [28]. Clearly, the presence of node-to-node overlap is undesirable, with overlaps reducing the degree of spatial localisation achieved, leading to a potential degradation in the performance of spatial queries.

In this work, I introduce a simple *node-inflation* strategy to construct spatial trees that are suitable for collections of AABB's. Given a collection generalised C, a spatial tree is built following a two-pass schedule, in which an initial tree is first constructed based on a point-wise interpretation of the collection C, with the tree then subsequently *inflated* to ensure that all AABB's are properly enclosed. In the first step, a tree $\mathcal{S}$ is built according to the top-down construction procedures outlined previously. Nodes are split according to a point-wise interpretation of the collection C, in which each entity $\mathbf{c}_i \in C$ is represented by the centroid of its associated AABB. This approach allows the standard point-wise node splitting techniques discussed previously to be applied without alteration. In the second step, the tree is inflated in a bottom-up fashion, in which the dimensions of the nodes of $\mathcal{S}$ are increased to ensure that each rectangle $R^k \in \mathcal{S}$ fully encloses all AABB's in its associated sub-collection $C^k \subseteq C$. This inflation is achieved by first ensuring that each leaf node $R^l \in \mathcal{S}$ fully encloses the AABB's within its associated sub-collection C^l. The tree is then traversed from the leaves up, performing a minimal inflation of the dimensions of each parent rectangle R^k, such that its children R_1^{k+1} and R_2^{k+1} are fully enclosed. See Figure 2.14 for an illustration of this inflation procedure. Since the inflation process consists of a simple traversal of the tree from leaf-to-root, it can be implemented efficiently in $O(n)$ time. Note that this inflation procedure can be used to augment both the 2^d- and KD-tree variants presented previously, facilitating the storage of general geometrical entities in both tree types. The tree inflation procedure is summarised in Algorithm 2.4.2.

As noted previously, the optimality of the resulting inflated tree can be degraded by the presence of node overlap, and a number of pathological situations can arise. Specifically, in cases where the dimensions of the entities in the collection C are relatively large compared to their local distribution of centroids, unacceptably large node overlaps can be induced. I mitigate this effect following the approach of Alliez, Tayeb and Wormser [1], in which and 'large' entities in the collection C are first subdivided into a series of sub-entities, such that the resulting collection is reasonably well distributed. Luckily, in the context of mesh generation, collections are generally automatically well-distributed,

Algorithm 2.4.2 Spatial Tree Inflation

function InflateTree($C, \gamma, \mathcal{S}$)
  Form a point-wise collection $\mathcal{C}$ of entity centroids.
  Call MakeTree($\mathcal{C}, \gamma, \mathcal{S}$) – forming a standard point-wise tree for the collection of entity centroids $\mathcal{C}$.
  Build a post-ordering $\mathcal{P}$ for the tree $\mathcal{S}$.
  for all ($k \in \mathcal{P}$) **do** ▷ {inflate tree}
    Call InflateNode($R^k, C^k, R^{k+1}, C^{k+1}$)
  end for
end function

function InflateNode($R^k, C^k, R^{k+1}, C^{k+1}$) ▷ {node inflation}
  if ($R^{k+1} = \emptyset$) **then** ▷ {leaf node}
    Form the AABB $\mathcal{B}$ for the entities in C^k.
    Inflate node R^k to enclose $\mathcal{B}$.
  else ▷ {non-leaf node}
    Inflate node R^k to enclose its children R_i^{k+1}.
  end if
end function

typically consisting of small lines and polygons that are well represented by compact axis-aligned bounding boxes. Additionally, I refer the reader to the more complex R*-tree strategy of Beckmann et al. [5, 6], for discussions of a more costly structure designed to find pseudo-optimal subdivisions for collections of AABB's directly. I do not pursue these methods further in this study, due to the excellent practical performance observed for the inflated AABB-tree type discussed previously.

2.4.4 Spatial Queries and Tree Traversal

The primary purpose of the 2^d-, KD-, and inflated AABB-tree types presented previously is the support of efficient spatial queries targeting the underlying collection of entities C stored in the tree. Typical queries include, for example: (i) determining the subset of entities in a collection that are included within, or intersect with a spatial halo $\mathcal{H}$, (ii) determining the entities in a collection that are closest to a given point $\mathbf{p} \in \mathbb{R}^d$, and (iii) testing the intersection of a given geometrical feature $\mathcal{F}$ with the entities in a collection. A naïve implementation of these, and other, spatial queries can be achieved by simply iterating over all entities in the collection C, explicitly evaluating the query predicates. Clearly, such methods result in $O(n)$ complexity per query, where $n = |C|$, and are thus unacceptably slow when the collection is large, as is often the case for the large-scale meshing problems investigated in this thesis. The performance of spatial queries can be significantly enhanced by exploiting the spatial locality induced by hierarchical trees.

Spatial queries can be implemented efficiently through the targeted traversal of an associated spatial tree $\mathcal{S}$. For example, the subset of entities $C_h \subseteq C$, enclosed or intersected by a spatial halo $\mathcal{H}$ can be determined efficiently via a top-down traversal of the tree $\mathcal{S}$, exploring only sub-trees that intersect with the region $\mathcal{H}$. This process exploits the bounding nature of the nodes of $\mathcal{S}$ – no entity within a sub-tree rooted at a node $R^k \in \mathcal{S}$ is enclosed or intersected by the halo $\mathcal{H}$ unless $R^k \cap \mathcal{H} \neq \emptyset$. Specifically, given a node $R^k \in \mathcal{S}$, the child nodes R_i^{k+1} are traversed only if they intersect with the halo region, such that $R_i^{k+1} \cap \mathcal{H} \neq \emptyset$. The traversal terminates once all suitable

Algorithm 2.4.3 Spatial Search

```
1: function SearchPred(C, 𝕊, 𝒫)
2:     Form axis-aligned halo ℋ for geometric predicate 𝒫.
3:     Call SearchTree(C, 𝕊, ℋ, Q_ℋ) to form queue of intersect-
       ing leaf nodes Q_ℋ.
4:     for all (k ∈ Q_ℋ) do                          ▷ {search local entities}
5:         for all (c_i ∈ C^k) do
6:             Evaluate predicate 𝒫(c_i) about local entity c_i.
7:         end for
8:     end for
9: end function

1: function SearchTree(C, 𝕊, ℋ, Q_ℋ)                 ▷ {tree traversal}
2:     Q_𝕊 ← R^0.
3:     while ((k ← Q_𝕊) ≠ ∅) do                      ▷ {search tree}
4:         if (R^k ⋂ ℋ ≠ ∅) then
5:             Traverse child nodes Q_𝕊 ← R_i^{k+1}.
6:             if (R^{k+1} = ∅) then                 ▷ {reached leaf node}
7:                 Push onto output Q_ℋ ← k.
8:             end if
9:         end if
10:    end while
11: end function
```

nodes have been visited, identifying a subset of the leaf nodes $R_h^l \in \mathcal{S}$ that intersect with, or are enclosed by the halo $\mathcal{H}$. A final pass then iterates through all entities in the sub-collections C_h^l associated with the subset of leaf nodes R_h^l, explicitly testing entities $\mathbf{c}_i \in C_h^l$ against the halo region $\mathcal{H}$. The traversal-based search process is summarised in Algorithm 2.4.3.

The efficiency of such a traversal-based approach is dependent on both the quality of the spatial tree $\mathcal{S}$ and the relative size of the halo region $\mathcal{H}$. In the best-case, in which the halo region $\mathcal{H}$ is sufficiently small, the traversal is limited to explore only a narrow sub-tree of $\mathcal{S}$, resulting in complexity that approaches $O(h)$ as $|\mathcal{H}| \to 0$. Furthermore, considering the high-quality spatial tree types discussed previously, logarithmic tree height is expected, leading to optimal $O(\log(n))$ query performance on average. In the worst-case, in which the halo region $\mathcal{H}$ is large compared to the dimensions of the tree $\mathcal{S}$, the number of nodes visited during the traversal increases. Recalling that the KD-tree variants presented previously ensure that the maximum number of nodes in the tree is guaranteed to be no larger than $O(n)$, such traversal based queries smoothly degenerate to simple linear methods of $O(n)$ complexity in the worst-case.

Tree-based algorithms designed for efficient spatial queries have been investigated extensively by a number of authors, and, as such, I do not provide extensive discussions here. I instead refer the reader to [39, 40] for further discussions and implementation details.

2.4.5 Implementation & Discussions

I have implemented the 2^d-, KD- and inflated AABB-tree types presented previously in a new spatial tree library – `LUMBERJACK`. While I restrict my attention to problems in $\mathbb{R}^2$ and $\mathbb{R}^3$ in this thesis, the implementations in `LUMBERJACK` have been designed to support

general collections of geometrical entities in $\mathbb{R}^d$. A number of associated spatial queries have also been implemented, including: (i) methods to locate the entities enclosed or intersected by a search window $\mathcal{H}$, and (ii) methods to find the intersection of a given geometrical feature $\mathcal{F}$ and a collection of entities. Support for general linear, rectangular and prismatic objects has been provided for both the search window and the intersection-based queries. Spatial queries are implemented following the efficient tree-based traversal methods outlined previously.

A series of comparative tests were used to assess the performance of the various spatial tree types. In Figure 2.15, results for the two-dimensional test cases are presented. A set of 3 point-wise geospatial data-sets were used to examine the effectiveness of both the 2^d- and KD-tree types, where the KD-tree incorporates the new node-shrinking strategy discussed in previous sections. These data-sets are derived from high-resolution coastline imagery and consist of collections of point-wise coordinates. All trees are refined to achieve a maximum node population threshold $\gamma = 16$. Analysis of Figure 2.15 shows that considerable difference in structure exists when comparing the 2^d- and KD-trees generated for each test case. Clear evidence of over-refinement is observed in the structure of the 2^d-trees, where a number of nodes in the 'interior' of each of the collections are clearly empty of data points. Conversely, the structure of the equivalent KD-trees shows significantly increased spatial adaptivity, with strong clustering occurring along the 'coastlines' in each data-set. Examining the size of the resulting trees, it is clear that the KD-trees are significantly smaller than the alternative 2^d-types for all test cases, with savings of approximately 20–40% achieved in each case.

In Figure 2.16, results for the three-dimensional test cases are presented, showing the associated 2^d- and AABB-trees generated for a series of discrete surface models. The model data is sourced from disparate application areas, including an engine rocker arm (ROCKER), the Stanford bunny (BUNNY) and a human hip bone (HIP). In each case, the set of triangular elements defining each surface is stored in the 2^d- and KD-trees as a collection of AABB's. All trees are refined to achieve a maximum node population threshold $\gamma = 32$. In all cases the *inflated* tree variants are used, ensuring that the simplicial entities are properly enclosed by the associated nodes in the trees. A visual analysis of the results shown in Figure 2.16 demonstrates that significant differences in the structure of the 2^d- and AABB-trees exist, as per the results in $\mathbb{R}^2$ discussed previously. It is clear that the flexible node splitting rules incorporated in the AABB-tree lead to enhanced spatial adaptivity, with the majority of nodes in such trees clustered near the 'surface' of the test objects. Examination of the resulting tree sizes illustrates that the AABB-trees significantly outperform the equivalent 2^d-trees in terms of total size, with savings of approximately 50% achieved in all cases.

2.5 Conclusions

In this chapter, I have reviewed a number of the theoretical concepts and geometrical constructs central to the development of the algorithms presented throughout this thesis. Specifically, I have introduced the notion of the Delaunay triangulation $\mathrm{Del}(X)$ – a sim-

Figure 2.15: Comparison of spatial trees for the NEWZEALAND, COAST and AUSTRALIA point-sets. The associated 2^d-trees are shown in the centre column, and the equivalent KD-trees are displayed in the rightmost column. Note that *node shrinking* was enabled for the KD-tress. The number of nodes in each tree, $|\mathcal{S}|$, is also shown for all cases. All tress were generated using a maximum node population threshold $\gamma = 16$.

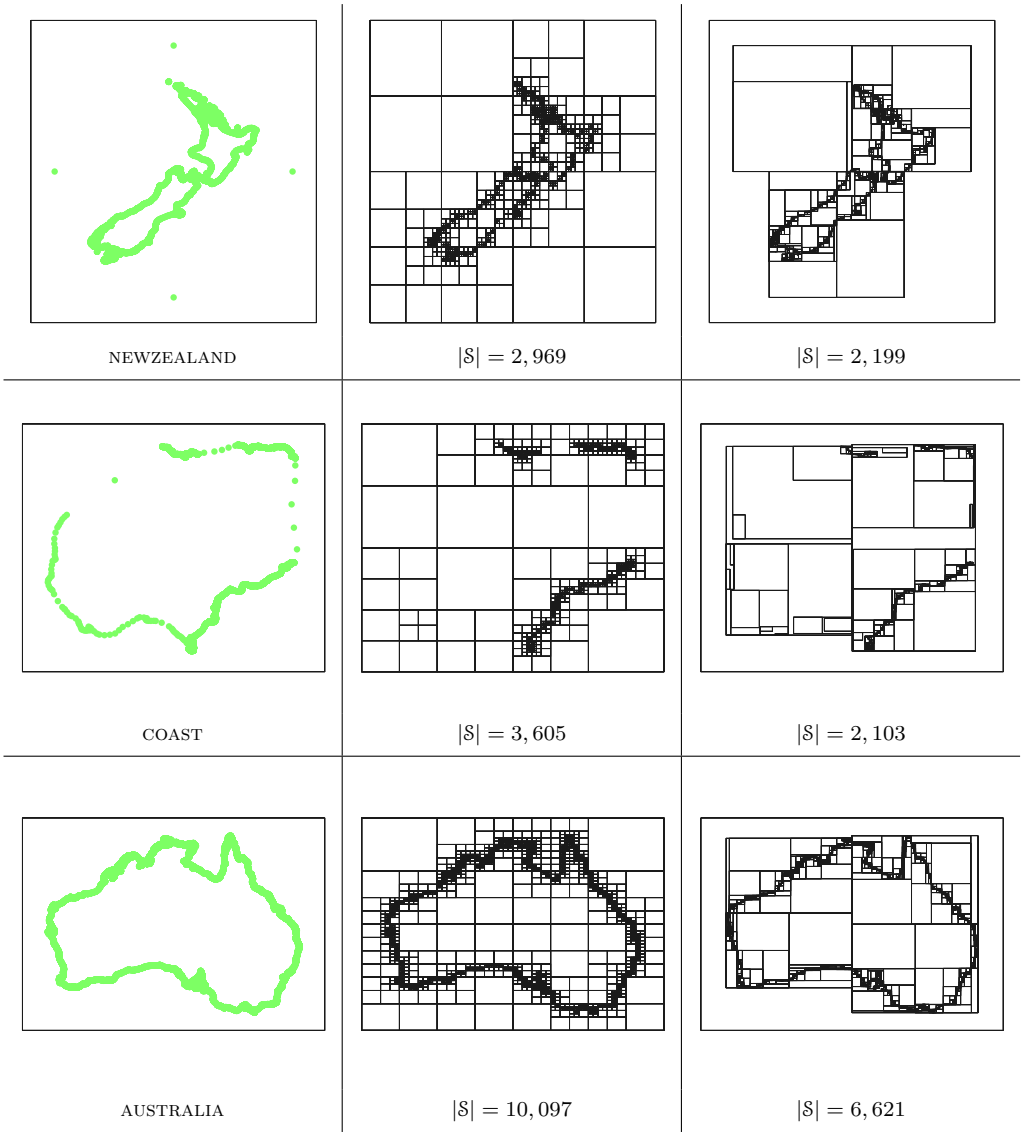
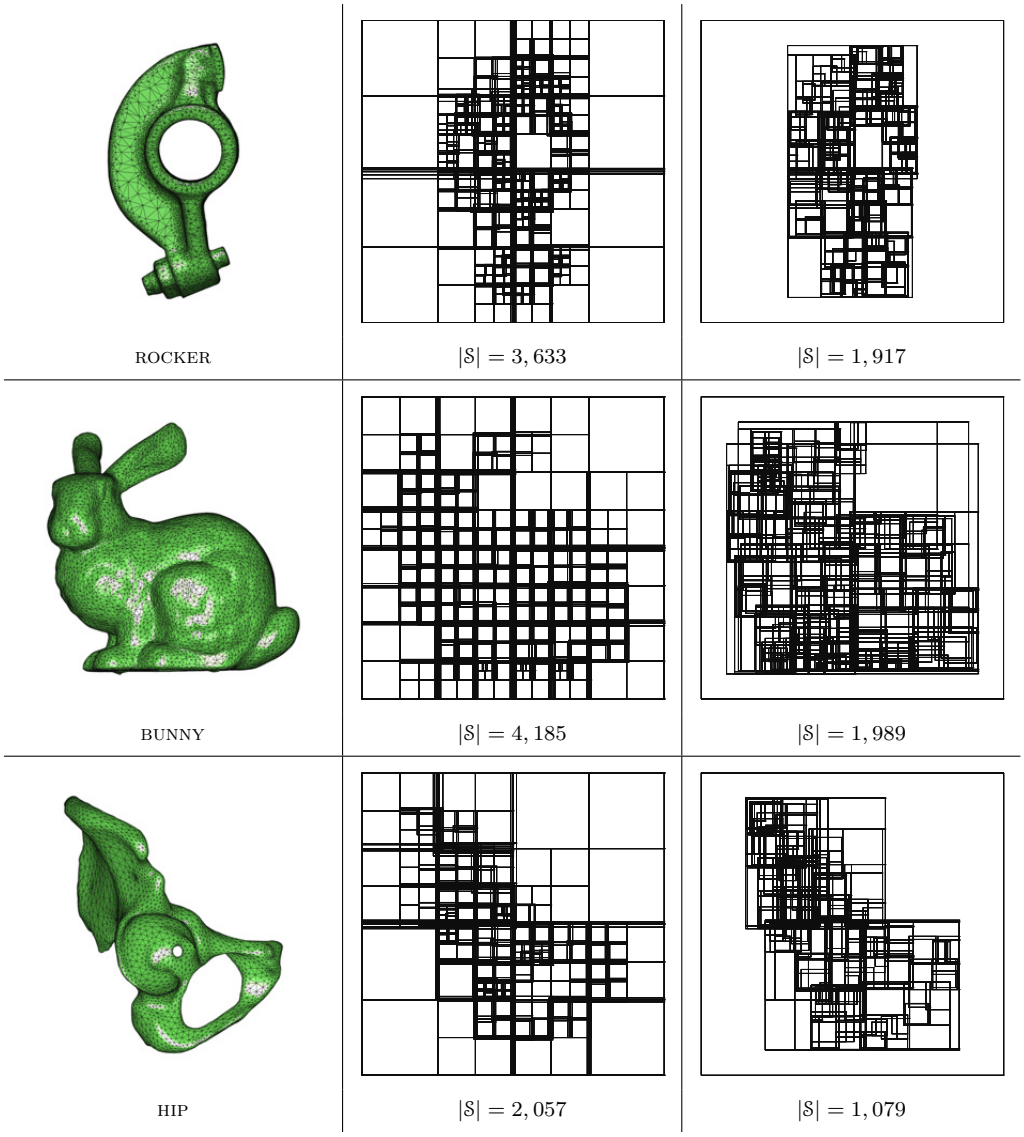


Figure 2.16: Comparison of spatial trees for the ROCKER, BUNNY and HIP manifold complexes. The associated 2^d-trees are shown in the centre column, and the equivalent AABB-trees are displayed in the rightmost column. Note that all trees are *inflated* variants. The number of nodes in each tree, $|\mathcal{S}|$, is also shown for all cases. All tress were generated using a maximum node population threshold $\gamma = 32$.



ROCKER	$\lvert\mathcal{S}\rvert = 3,633$	$\lvert\mathcal{S}\rvert = 1,917$
BUNNY	$\lvert\mathcal{S}\rvert = 4,185$	$\lvert\mathcal{S}\rvert = 1,989$
HIP	$\lvert\mathcal{S}\rvert = 2,057$	$\lvert\mathcal{S}\rvert = 1,079$

plicial tessellation for arbitrary point-sets $X \subset \mathbb{R}^d$. I have reviewed the properties and geometric optimality of such structures and have discussed their applicability to problems in mesh generation. Building upon this theoretical framework, I have implemented a set of efficient algorithms and data-structures, designed to facilitate the construction and incremental maintenance of such objects for problems in $\mathbb{R}^2$ and $\mathbb{R}^3$. A series of experimental studies were conducted, comparing the new Delaunay tessellation framework `TRIPOD` to the state-of-the-art triangulation packages `CGAL` and `QHULL`. These numerical experiments demonstrate that the performance of the `TRIPOD` framework is competitive with the best results achieved using existing packages. Such results confirm that the new `TRIPOD` library provides a high-quality basis for the development of efficient meshing algorithms.

In addition to Delaunay-based tessellations, I have also introduced a number of geometric structures based on hierarchical decompositions, whereby a collection of point-wise entities $C \subset \mathbb{R}^d$ is stored in a spatial tree $\mathcal{S}$, constructed via the recursive subdivision of bounding hyper-rectangles. In addition to a discussion of the well-known quadtree, octree and classical KD-tree objects, I have introduced a new KD-tree variant based on a series of new node-splitting techniques. Additionally, I have extended the existing spatial tree types to support the storage of generalised entities, such as lines and polyhedrons, through a new *node-inflation* technique. The various 2^d-, KD and AABB-tree types were implemented in a new spatial indexing library `LUMBERJACK`, and their performance was assessed using a series of comparative experimental studies. It was shown that the new KD- and AABB-tree types typically achieve a significant reduction in overall size when compared to conventional approaches, leading to improvements in computational efficiency. These results confirm that the new `LUMBERJACK` framework facilitates the type of efficient spatial search queries required when implementing high-performance meshing algorithms.

References

[1] Alliez, P., Tayeb, S., and Wormser, C. AABB Tree. Tech. rep., 2009.

[2] Amenta, N., Choi, S., and Rote, G. Incremental Constructions con BRIO. In *Proceedings of the nineteenth annual symposium on Computational geometry* (New York, NY, USA, 2003), SCG '03, ACM, pp. 211–219.

[3] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM (JACM) 45*, 6 (1998), 891–923.

[4] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. The Quickhull Algorithm for Convex Hulls. *TOMS 22*, 4 (1996), 469–483.

[5] Beckmann, N., Kriegel, H. P., Schneider, R., and Seeger, B. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 1990), SIGMOD '90, ACM, pp. 322–331.

[6] Beckmann, N., Kriegel, H. P., Schneider, R., and Seeger, B. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *SIGMOD Rec. 19*, 2 (May 1990), 322–331.

[7] Bentley, J. L. Multidimensional Binary Search Trees used for Associative Searching. *Commun. ACM 18*, 9 (Sept. 1975), 509–517.

[8] Boissonnat, J. D., Devillers, O., Pion, S., Teillaud, M., and Yvinec, M. Triangulations in CGAL. *Comput. Geom. Theory Appl. 22*, 1-3 (May 2002), 5–19.

[9] Boissonnat, J. D., and Oudot, S. Provably Good Surface Sampling and Approximation. In *ACM International Conference Proceeding Series* (2003), vol. 43, pp. 9–18.

[10] Boissonnat, J. D., and Oudot, S. Provably Good Sampling and Meshing of Surfaces. *Graphical Models 67*, 5 (2005), 405–451.

[11] Bowyer, A. Computing Dirichlet Tessellations. *The Computer Journal 24*, 2 (1981), 162–166.

[12] Cheng, S. W., Dey, T. K., and Levine, J. A. A Practical Delaunay Meshing Algorithm for a Large Class of Domains*. In *Proceedings of the 16th International Meshing Roundtable*, M. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 477–494.

[13] Cheng, S. W., Dey, T. K., and Ramos, E. A. Delaunay Refinement for Piecewise Smooth Complexes. *Discrete & Computational Geometry 43*, 1 (2010), 121–166.

[14] Cheng, S. W., Dey, T. K., Ramos, E. A., and Ray, T. Sampling and Meshing a Surface with Guaranteed Topology and Geometry. *SIAM journal on computing 37*, 4 (2007), 1199–1227.

[15] Cheng, S. W., Dey, T. K., and Shewchuk, J. R. *Delauay Mesh Generation.* Taylor & Francis, New York, 2013.

[16] Clarkson, K. L., and Shor, P. W. Applications of Random Sampling in Computational Geometry, II. *Discrete & Computational Geometry 4*, 1 (1989), 387–421.

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

[18] Devilliers, O. The Delaunay Heirarchy. *International Journal of Foundations of Computer Science 13*, 02 (2002), 163–180.

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

[20] Dwyer, R. A. A Faster Divide-and-conquer Algorithm for Constructing Delaunay Triangulations. *Algorithmica 2*, 1-4 (1987), 137–151.

[21] Dwyer, R. A. Higher-dimensional Voronoi Diagrams in Linear Expected Time. *Discrete & Computational Geometry 6*, 1 (1991), 343–367.

[22] Edelsbrunner, H., and Shah, N. R. Triangulating Topological Spaces. *International Journal of Computational Geometry & Applications 7*, 04 (1997), 365–378.

[23] Eppstein, D., Goodrich, M. T., and Sun, J. Z. Skip Quadtrees: Dynamic Data Structures for Multidimensional Point Sets. *International Journal of Computational Geometry & Applications 18*, 01n02 (2008), 131–160.

[24] Finkel, R., and Bentley, J. Quad trees a data structure for retrieval on composite keys. *Acta Informatica 4*, 1 (1974), 1–9.

[25] Frederick, C., Wong, Y., and Edge, F. Two-dimensional Automatic Mesh Generation for Structural Analysis. *International Journal for Numerical Methods in Engineering 2*, 1 (1970), 133–144.

[26] Freitag, L. A., and Ollivier-Gooch, C. Tetrahedral Mesh Improvement using Swapping and Smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[27] Guibas, L., and Stolfi, J. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transactions on Graphics (TOG) 4*, 2 (1985), 74–123.

[28] Guttman, A. *R-trees: A Dynamic Index Structure for Spatial Searching*, vol. 14. ACM, 1984.

[29] Hilbert, D. Ueber die stetige Abbildung einer Line auf ein Flächenstück. *Mathematische Annalen 38*, 3 (1891), 459–460.

[30] Klingner, B. M., and Shewchuk, J. R. Aggressive Tetrahedral Mesh Improvement. In *Proceedings of the 16th International Meshing Roundtable*, M. L. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 3–23.

[31] Lawson, C. L. Software for C^1 Surface Interpolation. In *Mathematical Software III*, J. R. Rice, Ed. Academic Press, New York, 1977, pp. 161–194.

[32] Lee, D. T., and Schachter, B. J. Two Algorithms for Constructing a Delaunay Triangulation. *International Journal of Computer & Information Sciences 9*, 3 (1980), 219–242.

[33] Maneewongvatana, S., and Mount, D. M. It's Okay to Be Skinny, If Your Friends Are Fat. In *Center for Geometric Computing 4th Annual Workshop on Computational Geometry* (1999).

[34] McLain, D. H. Two-dimensional Interpolation from Random Data. *The Computer Journal 19*, 2 (1976), 178–181.

[35] Mount, D. M., and Arya, S. ANN: Library for Approximate Nearest Neighbour Searching. In *Center for Geometric Computing 2nd Annual Fall Workshop on Computational Geometry*. 1998.

[36] Parthasarathy, V. N., Graichen, C. M., and Hathaway, A. F. A Comparison of Tetrahedron Quality Measures. *Finite Elements in Analysis and Design 15*, 3 (1994), 255 – 261.

[37] Rand, A., and Walkington, N. 3d Delaunay refinement of sharp domains without a local feature size oracle. In *Proceedings of the 17th International Meshing Roundtable*. Springer, 2008, pp. 37–54.

[38] Rand, A., and Walkington, N. Collars and intestines: Practical conforming Delaunay refinement. In *Proceedings of the 18th International Meshing Roundtable*. Springer, 2009, pp. 481–497.

[39] Samet, H. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company, 1990.

[40] Samet, H. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann. Elsevier/Morgan Kaufmann, 2006.

[41] Shewchuk, J. R. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry 18* (1996), 305–363.

[42] Shewchuk, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds., vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[43] Shewchuk, J. R. *Delaunay Refinement Mesh Generation*. PhD thesis, Pittsburg, Pennsylvania, 1997.

[44] Shewchuk, J. R. Sweep Algorithms for Constructing Higher-dimensional Constrained Delaunay Triangulations. In *Proceedings of the sixteenth annual symposium on Computational geometry* (2000), ACM, pp. 350–359.

[45] Shewchuk, J. R. Updating and Constructing Constrained Delaunay and Constrained Regular Triangulations by Flips. In *Proceedings of the nineteenth annual symposium on Computational geometry* (2003), ACM, pp. 181–190.

[46] Shewchuk, J. R. General-dimensional Constrained Delaunay and Constrained Regular Triangulations, I: Combinatorial Properties. *Discrete & Computational Geometry 39*, 1-3 (2008), 580–637.

[47] Watson, D. F. Computing the N-dimensional Delaunay Tessellation with Application to Voronoi Polytopes. *The computer journal 24*, 2 (1981), 167–172.

Chapter 3

Planar Mesh Generation

In this chapter, I present a new Frontal-Delaunay meshing algorithm for planar domains in $\mathbb{R}^2$. This algorithm is an extension of previous Delaunay-refinement and Frontal-Delaunay meshing strategies, designed to combine the high quality results achieved using advancing-front techniques with the provable theoretical bounds of Delaunay-refinement schemes. I review both the mechanics and theoretical guarantees associated with conventional Delaunay-refinement algorithms before introducing the new strategy. Exploiting ideas similar to those introduced by Rebay [15] and Üngör [20], I show that the use of 'off-centre' Steiner vertices, positioned along edges in the underlying Voronoi diagram, typically leads to an improvement in both the shape- and size-quality of the resulting tessellation. In addition to conventional shape-driven refinement methods, based on element radius-edge ratios, I show that a new size-optimal strategy can be realised by positioning off-centre vertices such that a local mesh size function is satisfied. The use of this sizing function to generate graded meshes adhering to user defined size constraints is also explored. I use a simple theoretical model to prove termination and convergence for the proposed algorithm. I investigate the performance of the new strategy experimentally, and undertake a series of comparative studies, contrasting the behaviour of the new algorithm with that of a typical Delaunay-refinement technique. I demonstrate that the new Frontal-Delaunay algorithm inherits many of the benefits of both Delaunay-refinement and advancing-front type methods, typically leading to the construction of very high quality triangulations in practice. Experiments are conducted using a range of complex benchmarks, verifying the robustness and practical performance of the proposed scheme.

3.1 Delaunay Refinement

Delaunay-refinement algorithms, as the name suggests, operate by incrementally introducing new Steiner vertices into an initially coarse Delaunay triangulation of the domain to be meshed. This *refinement* process continues until all elements in the mesh satisfy a set of shape and size constraints.

3.1.1 Ruppert's Algorithm

The first *provably-good* Delaunay-refinement algorithm is due to Ruppert [16, 17], who expanded on the earlier work of Chew [4]. Ruppert's algorithm takes as input a planar PLC $\mathcal{P}$ and an upper bound $\bar{\rho}$ on the maximum allowable element radius-edge ratios, returning a conforming Delaunay triangulation $\mathcal{T} \subseteq |\mathcal{P}|$, such that for all elements $\tau \in \mathcal{T}$ the radius-edge ratios are bounded, such that $\rho(\tau) \leq \bar{\rho}$. Ruppert showed that his algorithm is guaranteed to converge for $\bar{\rho} \geq \sqrt{2}$. Recalling that the radius-edge ratio is related to the element plane angles via $\rho(\tau) = \frac{1}{2}(\sin(\theta))^{-1}$, it is clear that Ruppert's algorithm guarantees a bound on element shape quality, ensuring that the plane angles lie in the range $20.7° \leq \theta(\tau) \leq 138.6°$. In the original algorithm, adjacent edges in $\mathcal{P}$ are required to meet at non-acute angles, although techniques to circumvent this restriction are presented in subsequent sections of this chapter.

Ruppert's algorithm begins by constructing a Delaunay triangulation $\mathcal{T} = \text{Del}(X)$ for the vertices $X \in \mathcal{P}$. In the next step, all edges of the PLC, $E \in \mathcal{P}$ are recovered by recursively splitting any edge $e \in E$ that is *encroached* by a vertex in X. It is said that an edge e is encroached if any vertex $\mathbf{x}_i \in X$, $\mathbf{x}_i \notin e$ lies within its closed diametric ball. Encroached edges are split via bisection. The main loop of the algorithm proceeds to incrementally refine any triangle $\tau \in \text{Del}(X)$ whose radius-edge ratio $\rho(\tau)$ exceeds the threshold $\bar{\rho}$. Triangles are typically refined by inserting their circumcentres $\mathbf{c}$, but if the insertion of $\mathbf{c}$ would lead to an edge e being encroached the edge e is bisected instead. The triangulation is modified upon each vertex insertion, such that $\mathcal{T} = \text{Del}(X)$ is always a Delaunay triangulation of the underlying vertex-set. Ruppert's algorithm terminates when the radius-edge ratios for all triangles are less than the threshold $\bar{\rho}$.

In addition to considerations of the element-wise radius-edge ratios, practical mesh generation algorithms based on Ruppert's algorithm typically also incorporate a size-driven refinement strategy. It is typical to supply a user-defined mesh size function $\bar{h}(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^+$ that expresses an upper bound on edge length as a function of position. In such cases, a high-quality triangulation can be generated using a slightly modified version of Ruppert's algorithm, in which both edges and triangles are refined to satisfy local sizing constraints, in addition to the radius-edge bounds described previously. Specifically, such a modification requires only that the convergence criteria be altered – ensuring that any edge $e \in E$ is refined if $\|\mathbf{e}\| \geq \alpha\bar{h}(\mathbf{x}_e)$, and that any triangle $\tau_i \in \text{Del}(X)$ is refined if $h(\tau) \geq \alpha\bar{h}(\mathbf{x}_\tau)$. Here, $\mathbf{x}_e$ and $\mathbf{x}_\tau$ are the edge midpoints and triangle circumcentres, respectively, and $\alpha \in \mathbb{R}^+$ is a 'safety-factor'[1]. The size of each triangle is expressed in a single-valued fashion[2], such that $h(\tau) = \sqrt{3}r$, where r is the radius of the associated circumcircle. The size of each edge is simply its Euclidean length, with $h(e) = \|\mathbf{e}\|$. Ruppert's algorithm, including modifications to support user-defined size constraints, is presented in Algorithm 3.1.1.

In practice, optimised implementations typically improve on the näive algorithm de-

[1] The scalar $\alpha = \frac{4}{3}$ in this study, to ensure that the mean element size does not, on average, undershoot the desired target size $\bar{h}(\mathbf{x}_\tau)$.

[2] The factor $\sqrt{3}$ represents the mapping between the edge-length and circumradius for an equilateral element.

Algorithm 3.1.1 Planar Delaunay Refinement.

```
1: function DelaunayPlanar(𝒫, ρ̄, h̄(x), 𝒯)
2:     Push all vertices x_i ∈ 𝒫 onto Del(X).
3:     Enqueue all 1- and 2-simplexes Q|_ℰ ← e ∈ Del(X) and Q|_𝒯 ←
       τ ∈ Del(X). Simplexes are enqueued if BadSimplex1(e) or
       BadSimplex2(τ) returns true.
4:     while (Q|_ℰ ≠ ∅) or (Q_𝒯 ≠ ∅) do                ▷ {main refinement loop}
5:         if (Q|_ℰ ≠ ∅) then
6:             Call RefineSimplex1(e ← Q|_ℰ)
7:         else
8:             Call RefineSimplex2(τ ← Q|_𝒯)
9:         end if
10:        for all (updated τ ∈ Del(X)) do
11:            Update Q|_ℰ and Q|_𝒯 to reflect changes in Del(X).
12:        end for
13:    end while
14:    return 𝒯 = Del(X)
15: end function
```

```
1: function RefineSimplex1(e)                          ▷ {edge refinement}
2:     Form diametric ball B(c, r) for edge e.
3:     Insert midpoint X ← c and update Del(X) ← X.
4: end function
```

```
1: function RefineSimplex2(τ)                          ▷ {area refinement}
2:     Form the Steiner point c for the simplex τ.
3:     if (c encroaches any edge e ∈ ℰ) then
4:         Call RefineSimplex1(e)
5:     else
6:         Insert Steiner point X ← c and update Del(X) ← X.
7:     end if
8: end function
```

```
1: function BadSimplex1(e)                             ▷ {termination criteria}
2:     return (e encroahed by any x_i ∈ Del(X)) or (h(e) > h̄(x_e))
3: end function
```

```
1: function BadSimplex2(τ)                             ▷ {termination criteria}
2:     return (ρ(τ) > ρ̄) or (h(τ) > h̄(x_τ))
3: end function
```

scribed previously in a number of important ways. Triangles are typically processed using a *priority-ordering*, sorted according to the element radius-edge ratios $\rho(\tau)$. The use of such a strategy ensures that the triangle with the *worst* radius-edge ratio is refined at each iteration of the main refinement loop. It is noted in [2, Chapter 6] that this 'worst-first' ordering can reduce the size of the output mesh $|\mathcal{T}|$ by up to 35% in some cases. Other important practical implementation details include: (i) using the current triangulation $\mathcal{T}$ as a geometric search structure when determining enclosing triangle and edge encroachment queries, and (ii) ensuring that only *internal* elements $\tau \in |\mathcal{P}|$ are refined, rather than processing all elements in $\text{Del}(X)$. In practice Ruppert's algorithm has been found to often significantly outperform its theoretical bounds, typically converging for up to $\theta_{\min} \simeq 33.8°$ for a wide range of inputs. See, for example, the previous studies [9, 18, 19, 20] for additional details and remarks.

3.1.2 Discussion

A simple theoretical model can be used to gain an understanding of the mechanics of Ruppert's algorithm, and to determine the conditions under which it is guaranteed to converge. Firstly, I introduce the so-called *Packing Lemma* – a key ingredient of the discussions that follow:

Lemma 3.1 (Packing Lemma)**.** *Let $D \subset \mathbb{R}^d$ be a bounded domain. Given a subset $X \subset D$, satisfying $\|\mathbf{u} - \mathbf{v}\| \geq \gamma$ for all pairs $\mathbf{u}, \mathbf{v} \in X$ and some scalar $\gamma \in \mathbb{R}^+$, the size of X is bounded, such that $|X| \leq \mu$ for some constant $\mu \in \mathbb{Z}^+$.*

The Packing Lemma states that any bounded set of points for which there exists a positive minimum separation distance must, consequently, be finite. As such, if it can be shown that a meshing algorithm preserves such a minimum separation length between its vertices, the Packing Lemma can be invoked to prove that the vertex-set is finite, and, as a result, that termination of the algorithm is guaranteed. I state Lemma 3.1 without proof – I refer interested readers to [2, Chapter 6] for full details.

A simple model for Ruppert's algorithm can be constructed by examining the effect of inserting the circumcentre $\mathbf{c}$ associated with a triangle $\tau \in \mathrm{Del}(X)$ into the underlying tessellation.

Proposition 3.1 (refinement, shape-quality)**.** *Let the element-wise radius-edge threshold $\bar{\rho} \geq \sqrt{2}$. Given a low-quality simplex $\tau \in \mathrm{Del}(X)$, for which $\rho(\tau) > \bar{\rho}$, the minimum edge length $\|\mathbf{e}_0\|$ is not decreased following the insertion of a new Steiner vertex located at the centre of the circumball associated with the element τ.*

Proof. Recalling the definition of the radius-edge ratio, shape-based refinement occurs when $r_d/\|\mathbf{e}_0\| \geq \bar{\rho}$, where r_d is the radius of the circumdisk $\mathrm{B}(\mathbf{c}, r)_d$ associated with the poor quality element τ. Considering the two modes of vertex insertion separately:

- Case (i): The circumcentre $\mathbf{c}_d$ *does not* encroach any constrained edge $e \in \mathcal{P}$. Noting that the simplex τ is Delaunay, it is clear that the adjacent vertices $\mathbf{x}_j \in \tau$ are the closest neighbours of the point $\mathbf{c}_d$, such that $\|\mathbf{x}_j - \mathbf{c}_d\| = r_d$. Following the insertion of $\mathbf{c}_d$, the new minimum edge length $\|\mathbf{e}_0{}'\|$ cannot be smaller than the circumradius r_d as a result. Rearranging the definition of the radius-edge ratio, the new minimum length can be expressed as $\|\mathbf{e}_0{}'\| \geq \bar{\rho}\,\|\mathbf{e}_0\|$, which is clearly non-decreasing for $\bar{\rho} \geq 1$.
- Case (ii): The circumcentre $\mathbf{c}_d$ *does* encroach at least one constrained edge $e \in \mathcal{P}$. Let $\mathrm{B}(\mathbf{c}, r)_e$ be the diametric ball associated with an encroached edge e, and let $\mathbf{x}_j \in e$ be the vertex that is positioned closest to the circumcentre $\mathbf{c}_d$. Noting that the simplex τ is Delaunay, it is clear that $\|\mathbf{x}_j - \mathbf{c}_d\| \geq r_d$. Rearranging the definition of the radius-edge ratio leads to $\|\mathbf{x}_j - \mathbf{c}_d\| \geq \bar{\rho}\,\|\mathbf{e}_0\|$. Due to encroachment, the constrained edge e is split by inserting a new vertex at $\mathbf{c}_e$, creating two short edges with $\|\mathbf{e}_0{}'\| = r_e$. In the limiting case, the circumcentre $\mathbf{c}_d$ is positioned on the circumference of the ball $\mathrm{B}(\mathbf{c}, r)_e$, forming a right-angle triangle such that $\|\mathbf{e}_0{}'\| = \left(\sqrt{2}/2\right)\|\mathbf{x}_j - \mathbf{c}_d\|$. Rearranging in terms of the existing minimum edge length leads to $\|\mathbf{e}_0{}'\| \geq \left(\sqrt{2}/2\right)\bar{\rho}\,\|\mathbf{e}_0\|$, which is non-decreasing for $\bar{\rho} \geq \sqrt{2}$.

□

In addition to shape-based refinement, it is important to also consider the effect of size-driven refinement strategies on the structure of the tessellation. Specifically, in such cases, a *high-quality* simplex $\tau \in \mathrm{Del}(X)$ with $\rho(\tau) \leq 1$ may be marked for further refinement if τ is in violation of the local element size constraints. In such cases, while it is tolerable to accept *some* reduction in minimum edge length, it is important to ensure that such a reduction is bounded in the worst-case, guaranteeing that the resulting sampling X remains *well-distributed*, without pathologically small edges.

Proposition 3.2 (refinement, grading)**.** *Given a refinable simplex $\tau \in \mathrm{Del}|_\Sigma(X)$, the minimum edge length $\|\mathbf{e}_0\|$ is decreased by a worst-case $O(1)$ factor following the insertion of a new Steiner vertex located at the centre of the associated circumball $\mathrm{B}(\mathbf{c}, r)_d$.*

Proof. Following the same reasoning presented in Proposition 3.1, the insertion of a new Steiner vertex $\mathbf{c}_d$, positioned at the centre of the associated circumball $\mathrm{B}(\mathbf{c}, r)_d$, modifies the local distribution of edge lengths. Specifically, the minimum edge length $\|\mathbf{e}_0{}'\|$ in the updated triangulation $\mathrm{Del}|_\Sigma(X \cup \{\mathbf{c}_d\})$ can be expressed in terms of the radius of the associated circumball, such that $\|\mathbf{e}_0{}'\| = \rho(\tau) r_0$. Noting that $\rho(\tau)$ achieves a minimum value when the simplex τ is equilateral, such that $\rho(\tau) = 1/\sqrt{3}$, it is clear that the minimum edge length is reduced by an $O(1)$ factor of $1/\sqrt{3}$ in the worst-case. □

Guarantees on the worst-case behaviour of the refinement operators can be used to study the termination and convergence of the algorithm.

Proposition 3.3 (termination)**.** *Given a polygonal domain $\mathcal{P}$, a radius-edge threshold $\bar{\rho} \geq \sqrt{2}$, and a positive mesh size function $\bar{h}(\mathbf{x}) > 0$ defined for all $\mathbf{x} \in |\mathcal{P}|$, the Delaunay-refinement algorithm (3.1) terminates in a finite number of steps.*

Proof. The Delaunay-refinement algorithm (3.1) refines any simplex $\tau \in \mathrm{Del}(X)$ if: (i) it is of poor shape quality, such that $\rho(\tau) \geq \bar{\rho}$, or (ii) it is too large, such that it exceeds local mesh size constraints, such that $h(\tau) \geq \alpha \bar{h}(\mathbf{x}_\tau)$.

- Case (i): Given a sufficiently large radius-edge threshold $\bar{\rho} \geq \sqrt{2}$ it is known, via Proposition 3.1, that refinement does not lead to a reduction in minimum edge length. Shape-based refinement therefore preserves the minimal edge length $\|\mathbf{e}_0\|_k$, associated with either the initial sampling $X_0 \in \mathcal{P}$, or the insertion of some previous vertex $\mathbf{x}_k \in X$ due to size-driven refinement.
- Case (ii): Given that the mesh size function $\bar{h}(\mathbf{x})$ is positive, such that $\bar{h}(\mathbf{x}) \geq \bar{h}_0$ for some $\bar{h}_0 \in \mathbb{R}^+$, size-driven refinement is *declined* once the local element size is sufficiently small. Specifically, considering a limiting case, in which $h(\tau) = \sqrt{3} r = \alpha \bar{h}_0$, where r is the radius of the associated circumball, Proposition 3.2 states that the minimum edge length is reduced by a factor of $1/\sqrt{3}$ in the worst-case. Rearranging the previous expressions, the minimum edge length is shown to be bounded above $(\alpha/3)\bar{h}_0$.

Given the individual bounds on minimum edge length, the overall Delaunay-refinement algorithm (3.1) is known to preserve some positive minimum vertex separation

distance $d = \min\big(\|\mathbf{e}_0\|_k, (\alpha/3)\bar{h}_0\big)$, and, via the Packing Lemma 3.1, is guaranteed to produce a point-wise sampling X of finite size. It is therefore clear that the algorithm inserts a bounded number of Steiner vertices and is guaranteed to terminate in a finite number of steps as a consequence. □

A proof of finite termination leads directly to a number of useful auxiliary guarantees and bounds on both the nature and the quality of the output tessellation. Adopting the conventional terminology, meshes generated using the Delaunay-refinement algorithm presented in Section 3.1 can be considered to be 'provably-good'.

Corollary 3.1 (convergence)**.** *Given a polygonal domain $\mathcal{P}$, a radius-edge threshold $\bar{\rho} \geq \sqrt{2}$, and a positive mesh size function $\bar{h}(\mathbf{x}) > 0$ defined for all $\mathbf{x} \in |\mathcal{P}|$, the tessellation $\mathcal{T} = \mathrm{Del}(X)$ generated by the Delaunay-refinement algorithm (3.1) satisfies all imposed constraints. Specifically, for all simplexes τ in the output $\mathcal{T}$: (i) all radius-edge ratios are bounded, such that $\rho(\tau) < \bar{\rho}$, and (ii) all element sizes are bounded, such that $h(\tau) < \alpha\bar{h}(\mathbf{x}_\tau)$, where $\mathbf{x}_\tau$ is the centre of the associated circumball $\mathrm{B}(\mathbf{c}, r)_d$ and $\alpha \in \mathbb{R}^+$.*

Proof. Algorithm (3.1) maintains a queue of 'bad' elements throughout the refinement process, containing any simplexes $\tau \in \mathrm{Del}(X)$ that are in violation of one or more local constraints. Specifically, a given triangle $\tau \in \mathrm{Del}(X)$ is enqueued if: (i) $\rho(\tau) \geq \bar{\rho}$, or (ii) if $h(\tau) \geq \alpha\bar{h}(\mathbf{x}_\tau)$. Given that the algorithm is known to terminate, it is clear that the refinement queue must become empty after a finite number of steps and that all entities in the resulting triangulation $\mathrm{Del}(X)$ satisfy the requisite radius-edge ratio and element size constraints as a consequence. □

3.2 Frontal-Delaunay Methods

Frontal-Delaunay algorithms are a hybridisation of advancing-front and Delaunay-refinement techniques, in which a Delaunay triangulation is used to define the topology of a mesh while new Steiner vertices are inserted in a manner consistent with advancing-front techniques. In practice, such methods have been observed to produce very high-quality meshes, inheriting the smooth, semi-structured vertex placement of pure advancing-front methods and the optimal mesh topology of Delaunay-based techniques. Early Frontal-Delaunay meshing algorithms are due to Rebay [15], Müller, Roe and Deconinck [10] and Mavriplis [8] who built upon advancing-front frameworks to generate high-quality meshes for problems in computational fluid dynamics. More recently Erten and Üngör [6, 20] have shown that similar methods can instead be developed through modifications to the Delaunay-refinement framework.

3.2.1 Off-centres

The Frontal-Delaunay algorithm presented in this study is primarily based on ideas introduced by Rebay, who, in [15], developed a Frontal-Delaunay algorithm in which new vertices are positioned along edges in the associated Voronoi diagram. Rebay's approach was based on size-driven refinement, positioning new Steiner vertices along edges $e \in$

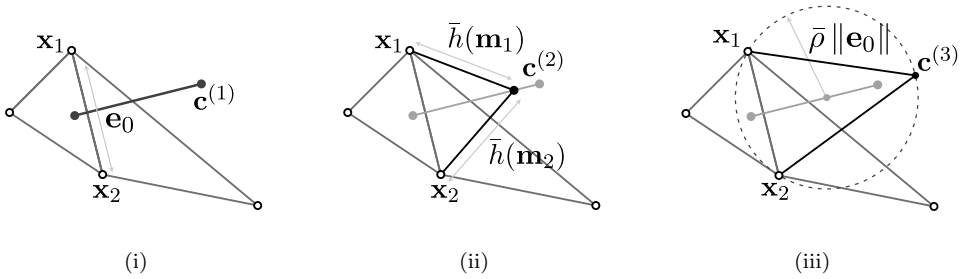
$\text{Vor}(X)$ in order to satisfy a prescribed mesh size function $h(\mathbf{x})$ – a strategy broadly consistent with conventional advancing-front techniques. While Rebay's algorithm uses a Delaunay triangulation $\mathcal{T} = \text{Del}(X)$ to define the topology of the mesh throughout the refinement procedure, consistent with conventional Delaunay-refinement algorithms, it is still fundamentally an advancing-front scheme – bereft of guarantees on the element shape quality $\rho(\tau)$ and $\theta(\tau)$.

In contrast, Üngör and Erten have approached the problem from the flip-side, introducing the notion of *generalised* Steiner vertices for Delaunay-refinement methods. Their so-called 'off-centre' vertices are points other than triangle circumcentres, lying along edges in the associated Voronoi diagram, as per Rebay. In [20], Üngör showed that when refining a triangle $\tau \in \text{Del}(X)$ its off-centre should be positioned, if possible, such that the new triangle τ' adjacent to the shortest edge $e_0 \in \tau$ satisfies the desired radius-edge constraint, such that $\rho(\tau) \leq \bar{\rho}$. Importantly, Üngör demonstrated that such a strategy typically leads to an improvement in the performance of the Delaunay-refinement algorithm in practice, significantly reducing the size of the output $|\mathcal{T}|$ when the threshold $\bar{\rho}$ is restrictive. Üngör also extended the theoretical guarantees associated with Ruppert's Delaunay-refinement algorithm to his off-centre technique and showed that the same shape quality bounds are satisfied. Üngör's off-centre refinement algorithm is based on Ruppert's Delaunay-refinement framework directly, involving only the modification of the point-placement scheme used to refine poor quality triangles. The use of similar *generalised* refinement strategies has also been explored by Chernikov, Chrisochoides and Foteinos in [3, 7].

In this study, I consider the use of off-centre Steiner vertices to simulate the vertex placement strategy of a conventional advancing-front approach, while also preserving the framework of a Delaunay-refinement meshing algorithm. The aim of such a strategy is to recover the high element qualities and smooth, semi-structured meshes generated by frontal methods in practice, while inheriting the guarantees and provable bounds of Delaunay-refinement based strategies. Advancing-front algorithms typically incorporate a mesh size function $\bar{h}(\mathbf{x})$ – a function $f : \mathbb{R}^2 \to \mathbb{R}^+$ defined over the domain to be meshed, where $\bar{h}(\mathbf{x})$ represents the desired edge length $\|\mathbf{e}\|$ at a point $\mathbf{x} \in |\mathcal{P}|$. This mesh size function typically incorporates size constraints dictated by the both the user and the geometry of the domain to be meshed. The construction of appropriate mesh size functions will be discussed in further detail in Section 3.3, but for now, it is sufficient to note that $\bar{h}(\mathbf{x})$ is a g-Lipschitz continuous function defined at all points on the domain to be meshed.

The proposed Frontal-Delaunay algorithm developed in this study is an extension of Ruppert's Delaunay-refinement algorithm presented in Section 3.1, modified to use off-centre rather than circumcentre-based refinement schemes. The basic framework of the algorithm is consistent with the Delaunay-refinement algorithm described previously, in which an initially coarse triangulation of the domain $\mathcal{P}$ is refined through the introduction of additional Steiner vertices $\mathbf{x}_i \in \mathcal{P}$ until all element shape and size constraints are satisfied. The constraints satisfied by the Frontal-Delaunay algorithm are identical to those discussed previously for the modified Delaunay-refinement scheme, with upper

Figure 3.1: Off-centre constructions for a triangle τ, illustrating (i) the segment of the Voronoi diagram associated with the short edge $\mathbf{e}_0 \in \tau$, (ii) placement of the size-optimal vertex $\mathbf{c}^{(2)}$ such that local size constraints $\bar{h}(\mathbf{x}_\tau)$ are enforced, and (iii) placement of a shape-optimal vertex $\mathbf{c}^{(3)}$ such that the required radius-edge ratio $\bar{\rho}$ is satisfied.



(i) (ii) (iii)

bounds on the radius-edge ratio $\bar{\rho}$ and element size $\bar{h}(\mathbf{x}_\tau)$ all required to be satisfied for convergence. I refer the reader to Algorithm 3.1.1 for a detailed summary of the method.

3.2.2 Point-placement Strategy

Given a triangle $\tau \in \mathrm{Del}(X)$ marked for refinement, the new Steiner vertex introduced to eliminate τ is an off-centre, constructed based on local size and shape constraints. Adopting the *generalised* off-centre framework introduced by Üngör, the 'ideal' location of the off-centre $\mathbf{c}$ for the given element τ is based on a consideration of the isosceles triangle σ formed about the short edge $\mathbf{e}_0 \in \tau$. I consider the locally-optimal placement of three points, $\mathbf{c}^{(1)}$, $\mathbf{c}^{(2)}$ and $\mathbf{c}^{(3)}$, designed to ensure that σ satisfies both local shape and size constraints. Type I vertices, $\mathbf{c}^{(1)}$, are equivalent to conventional element circumcentres, and are used to satisfy constraints on the element radius-edge ratios. Type II vertices, $\mathbf{c}^{(2)}$, are *size-optimal* points, designed to satisfy element sizing constraints in a locally optimal fashion. Type III points, $\mathbf{c}^{(3)}$, are Üngör's off-centre vertices, and are designed to reduce the number of Steiner vertices required to satisfy the desired element radius-edge ratio. The final off-centre $\mathbf{c}$ is chosen as the point $\mathbf{c}^{(i)}$ that satisfies local constraints in a worst-case fashion. Point-placement diagrams for the generalised off-centre schemes are shown in Figure 3.1.

Given a *refinable* element $\tau \in \mathrm{Del}(X)$, the size-optimal Type II vertex $\mathbf{c}^{(2)}$ is placed following an approach similar to that introduced by Rebay. The point $\mathbf{c}^{(2)}$ is positioned along the Voronoi segment[1] $\mathbf{v} \in \mathrm{Vor}(X)$ associated with the short edge $\mathbf{e}_0 \in \tau$, such that the size of the new triangle $h(\sigma)$ satisfies local constraints. Specifically, the altitude of the triangle is calculated to ensure that the two new edges of σ are not too long, such that $\|\mathbf{e}_1\| \simeq \bar{h}(\mathbf{m}_1)$ and $\|\mathbf{e}_2\| \simeq \bar{h}(\mathbf{m}_2)$, where the $\mathbf{m}_i$'s are the edge midpoints. Each constraint can be solved for an associated altitude

$$a_i^{(2)} = \left(\bar{h}(\mathbf{m}_i)^2 - \|\tfrac{1}{2}\mathbf{e}_0\|^2\right)^{\frac{1}{2}}. \tag{3.1}$$

[1]The Voronoi segment $\mathbf{v}$ associated with an edge $\mathbf{e} \in \mathcal{T}$ is the line segment connecting the circumcentres of the triangles τ_1, τ_2 adjacent to $\mathbf{e}$. Their intersection $\mathbf{e} \cap \mathbf{v}$ is $\mathbf{m}$, the midpoint of the edge $\mathbf{e}$.

Given the pair of altitudes, the position of the size-optimal point $\mathbf{c}^{(2)}$ can be expressed as

$$\mathbf{c}^{(2)} = \mathbf{m}_0 + \tfrac{1}{2}\left(a_1^{(2)} + a_2^{(2)}\right)\hat{\mathbf{v}} \tag{3.2}$$

where $\mathbf{m}_0$ is the midpoint of the short edge $\mathbf{e}_0$ and $\hat{\mathbf{v}}$ is the *frontal* unit direction vector associated with the Voronoi edge $\mathbf{v}$. Note that, for non-uniform $\bar{h}(\mathbf{x})$, this expression is non-linear, with the altitudes $a_i^{(2)}$ depending on the evaluation of the mesh size function at the edge midpoints $\bar{h}(\mathbf{m}_i)$ and visa-versa. In practice, since $\bar{h}(\mathbf{x})$ is Lipschitz smooth, a simple iterative predictor-corrector procedure is sufficient to solve these expressions approximately.

The *shape-optimal* vertex $\mathbf{c}^{(3)}$ is placed following the procedure of Üngör. The point $\mathbf{c}^{(3)}$ is placed along the Voronoi segment $\mathbf{v} \in \mathrm{Vor}(X)$ associated with the short edge $\mathbf{e}_0 \in \tau$, such that the shape of the new triangle σ satisfies $\rho(\sigma) \leq \bar{\rho}$. Setting $\rho(\sigma) = \bar{\rho}$ leads to a solution for the *altitude* of σ

$$a^{(3)} = \tfrac{1}{2}\|\mathbf{e}_0\|\left(\tan(\tfrac{1}{2}\theta_{\min})\right)^{-1} \tag{3.3}$$

where $\theta_{\min}$ = is the minimum angle associated with the radius-edge constraint $\bar{\rho}$. The position of the shape-optimal point $\mathbf{c}^{(3)}$ is then expressed as

$$\mathbf{c}^{(3)} = \mathbf{m}_0 + a^{(3)}\,\hat{\mathbf{v}} \tag{3.4}$$

where $\mathbf{m}_0$ is the midpoint of the short edge $\mathbf{e}_0$ and $\hat{\mathbf{v}}$ is the unit direction vector associated with the Voronoi edge $\mathbf{v}$.

Using the size-optimal Type II point $\mathbf{c}^{(2)}$, the shape-optimal Type III point $\mathbf{c}^{(3)}$ and the Type I point $\mathbf{c}^{(1)}$, the final position of the refinement point $\mathbf{c}$ for the element τ is calculated. The point $\mathbf{c}$ is selected to satisfy the *limiting* local constraints, setting

$$\mathbf{c} = \begin{cases} \mathbf{c}^{(2)}, & \text{if } \left(d^{(2)} \leq d^{(1)}\right),\ \left(d^{(2)} \leq d^{(3)}\right) \text{ and } \left(d^{(2)} \geq \tfrac{1}{2}\|\mathbf{e}_0\|\right), \\ \mathbf{c}^{(3)}, & \text{if } \left(d^{(3)} \leq d^{(1)}\right), \\ \mathbf{c}^{(1)}, & \text{otherwise} \end{cases} \tag{3.5}$$

where the $d^{(i)} = \|\mathbf{c}^{(i)} - \mathbf{m}_0\|$ are distances from the midpoint of the frontal edge $\mathbf{e}_0$ to the Type I, Type II and Type III vertices, respectively. The cascading selection criteria is designed to ensure that the refinement scheme smoothly degenerates to that of a conventional circumcentre-based Delaunay-refinement strategy in limiting cases, while using locally size- or shape-optimal points where possible. Specifically, the conditions $d^{(2,\,3)} \leq d^{(1)}$ guarantee that $\mathbf{c}$ lies no further from the frontal edge $\mathbf{e}_0$ than the centre of the circumball of τ. Additionally, the condition $d^{(2)} \geq \tfrac{1}{2}\|\mathbf{e}_0\|$ ensures that the diametric ball of the edge $\mathbf{e}_0$ remains empty. Such behaviour guarantees that the size-optimal scheme is only selected when $\mathbf{e}_0$ is sufficiently small with respect to the local mesh size function – ensuring that $\mathbf{e}_0$ is a good *frontal* edge candidate in the context of a conventional advancing-front scheme. In all other cases, a shape-based strategy, guaranteed to reduce element radius-edge ratios, is selected.

3.2.3 Discussion

The mechanics of the Frontal-Delaunay algorithm are similar to those of the Delaunay-refinement scheme presented in Section 3.1. Specifically, since the Frontal-Delaunay algorithm involves a modification to the underlying point-placement strategy only, it inherits much of the same theoretical framework as the preceding Delaunay-refinement scheme. Differences in the point-placement strategies necessitate that a number of the propositions presented in Section 3.1 be re-evaluated for the new algorithm.

Proposition 3.4 (refinement, shape-quality)**.** *Let the element-wise radius-edge threshold $\bar{\rho} \geq \sqrt{2}$. Given a low-quality simplex $\tau \in \mathrm{Del}(X)$, for which $\rho(\tau) > \bar{\rho}$, the minimum edge length $\|\mathbf{e}_0\|$ is not decreased following either: (i) the insertion of a new Type I Steiner vertex, located at the centre of the associated circumball $\mathrm{B}(\mathbf{c}, r)$, or (ii) the insertion of a new Type III Steiner vertex, positioned on an adjacent edge in the associated Voronoi diagram.*

Proof. Considering the two shape-driven vertex insertion strategies separately:

- Case (i): The bounds associated with the insertion of Type I Steiner vertices are presented in detail in Proposition 3.1, and I do not reproduce them here in full as a result. Specifically, recall that the minimum edge length is guaranteed to be non-decreasing given $\bar{\rho} \geq \sqrt{2}$.
- Case (ii): Type III Steiner vertices are positioned on an edge in the Voronoi diagram $\mathbf{v} \in \mathrm{Vor}(X)$, associated with the short edge $e_0 \in \tau$. Based on the properties of the Voronoi diagram, the edge $\mathbf{v}$ intersects e_0 at its midpoint. The Steiner vertex $\mathbf{c}^{(3)}$ is positioned to ensure that an isosceles triangle σ, formed between the existing vertices $\mathbf{x}_{i,j} \in e_0$ and $\mathbf{c}^{(3)}$, satisfies the desired shape-quality. Recalling that the point $\mathbf{c}^{(3)}$ lies on an adjacent edge in the Voronoi diagram, it is clear that the existing vertices $\mathbf{x}_{i,j} \in e_0$ are its closest neighbours. Considering the geometry of the triangle σ, the length of the new edges can be expressed in terms of the base length, such that $\|\mathbf{e}_0{}'\| = \frac{1}{2}\big(\sin\big(\frac{1}{2}\theta_{\min}\big)\big)^{-1}\|\mathbf{e}_0\|$, where $\theta_{\min}$ is the enclosed angle at the apex of σ. Clearly, for $\theta_{\min} \leq 60°$, the existing minimum edge length $\|\mathbf{e}_0\|$ is preserved. Such a limit corresponds to an equivalent bound on the radius-edge ratio, such that $\bar{\rho} \geq 1/\sqrt{3}$.

Overall, it is clear that the insertion of Type I vertices is a limiting case, and that, as a result, any point-placement scheme based on the insertion of both Type I and Type III Steiner vertices requires $\bar{\rho} \geq \sqrt{2}$ to ensure that the minimum edge length is preserved. Such a bound is consistent with the threshold given in Proposition 3.1 for the standard Delaunay-refinement scheme. □

Proposition 3.5 (refinement, grading)**.** *Given a refinable simplex $\tau \in \mathrm{Del}|_{\Sigma}(X)$, the minimum edge length $\|\mathbf{e}_0\|$ is decreased by an $O(1)$ factor in the worst-case following: (i) the insertion of a new Type I Steiner vertex located at the centre of the associated circumball $\mathrm{B}(\mathbf{c}, r)_d$, or (ii) the insertion of a new Type II Steiner vertex, positioned on an adjacent edge in the associated Voronoi diagram.*

Proof. Considering the two size-driven vertex insertion strategies separately:

- Case (i): The bounds associated with the insertion of Type I Steiner vertices are presented in detail in Proposition 3.2, and I do not reproduce them here in full as a result. Specifically, recalling that in the worst-case, given an equilateral triangle τ, Type I refinement results in a new minimum edge length equal to the radius of the associated circumball, such that the edge length is bounded above $(1/\sqrt{3})\|\mathbf{e}_0\|$.
- Case (ii): Type II Steiner vertices are positioned on an edge in the Voronoi diagram $\mathbf{v} \in \text{Vor}(X)$, associated with the short edge $e_0 \in \tau$. Based on the properties of the Voronoi diagram, the edge $\mathbf{v}$ intersects e_0 at its midpoint. The Steiner vertex $\mathbf{c}^{(2)}$ is positioned to ensure that an isosceles triangle σ, formed between the existing vertices $\mathbf{x}_{i,j} \in e_0$ and $\mathbf{c}^{(2)}$, satisfies the desired local size constraints. In the worst-case, when $\bar{h}(\mathbf{x}) \leq \|\mathbf{e}_0\|$, the new vertex $\mathbf{c}^{(2)}$ is positioned at the intersection of $\mathbf{v}$ and the diametric ball associated with the edge e_0. Such a strategy ensures that the new edge length is bounded above $(1/\sqrt{2})\|\mathbf{e}_0\|$.

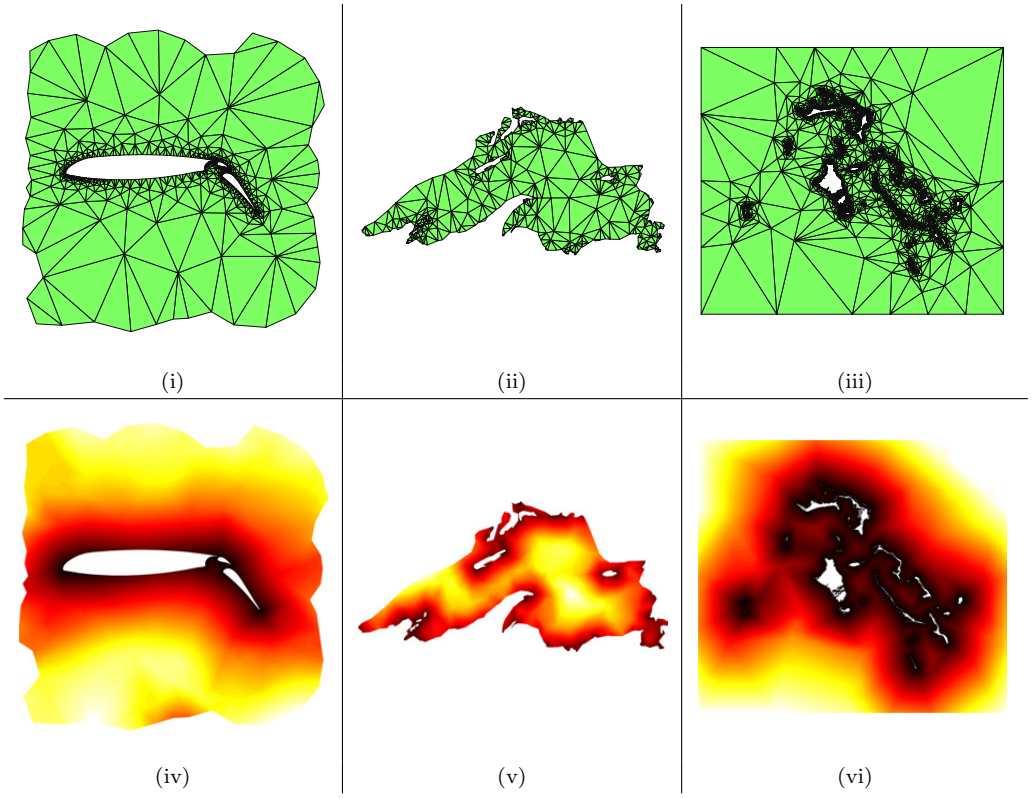
Overall, it is clear that the insertion of Type I vertices is a limiting case, and that, as a result, any point-placement scheme based on the insertion of both Type I and Type II Steiner vertices reduces minimum edge length by a factor of $(1/\sqrt{3})$ in the worst-case, consistent with the bounds given in Proposition 3.2 for the standard Delaunay-refinement scheme. □

Importantly, the bounds given in Propositions 3.4 and 3.5 are identical to those derived for the conventional Delaunay-refinement algorithm presented in Section 3.1, showing that the behaviour of the new Frontal-Delaunay and Delaunay-refinement algorithms is consistent in the worst-case. Specifically, the bounds stated in Propositions 3.4 and 3.5 show that the new Frontal-Delaunay algorithm is guaranteed to: (i) terminate in a finite number of steps, and (ii) converge to a result satisfying all element shape- and size-driven constraints. A proof of these statements is identical to that presented previously for the Delaunay-refinement algorithm (3.1), via Proposition 3.3 and Corollaries 3.1 and is therefore not reproduced in full here.

3.3 Mesh Size Functions

The construction of high-quality mesh size functions is an important aspect of the Frontal-Delaunay algorithm presented in Section 3.2. A good mesh size function $\bar{h}(\mathbf{x})$ incorporates sizing constraints imposed by both the user and the geometry of the domain to be meshed. These contributions can be considered via two separate size functions, where $\bar{h}_u(\mathbf{x})$ represents user-defined sizing information and $\bar{h}_g(\mathbf{x})$ encapsulates sizing constraints dictated by the geometry of the domain $\mathcal{P}$. In this study, I require that both $\bar{h}_u(\mathbf{x})$ and $\bar{h}_g(\mathbf{x})$ be piecewise linear functions defined on a supporting triangular complex $\mathcal{S}$ and that the supporting complex *cover* the domain to be meshed, such that $|\mathcal{P}| \subseteq |\mathcal{S}|$. These restrictions are typical of methods used in existing mesh generation algorithms. Construction of appropriate user-defined functions $\bar{h}_u(\mathbf{x})$ is highly problem specific and I

Figure 3.2: Construction of the mesh size functions for the AIRFOIL, LAKE and ISLAND planar meshing problems, showing the sparse supporting complexes $\mathcal{S}_g$ in (i)–(iii), and contours of the geometric mesh size functions $\bar{h}_g(\mathbf{x})$ in (iv)–(vi).



do not attempt to canvas individual methods here. As a general comment, two typical choices for $\bar{h}_u(\mathbf{x})$ often arise in practice. In the first case, a uniform maximum size is required, leading to the obvious choice $\bar{h}_u(\mathbf{x}) = \gamma,\ \gamma \in \mathbb{R}^+$. The second case addresses solution adaptive re-meshing, in which a piecewise function $\bar{h}_u(\mathbf{x})$ is derived from the characteristics of a numerical solution $\psi(\mathbf{x})$, represented on an existing mesh $\mathcal{T}^k$. Typically, this size data is used to obtain a new mesh $\mathcal{T}^{k+1}$ that better resolves $\psi(\mathbf{x})$.

3.3.1 Initial Size Estimates

Before outlining the procedure used to construct the geometric mesh size function $\bar{h}_g(\mathbf{x})$, I introduce the so-called *local feature size* for a polygonal PLC.

Definition 3.1 (local feature size)**.** The local feature size $\text{lfs}(\mathbf{x})$ of a polygonal domain $\mathcal{P}$ is a function $f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^+$, where $f(\mathbf{x})$ is the radius of the smallest disk centred at $\mathbf{x}$ intersecting two non-adjacent edges in $\mathcal{P}$.

Ideally, the local feature size of $\mathcal{P}$ could be used directly as a geometric size function, such that $\bar{h}_g(\mathbf{x}) = \text{lfs}(\mathbf{x})$. In practice, since direct evaluation of $\text{lfs}(\mathbf{x})$ is expensive, I construct $\bar{h}_g(\mathbf{x})$ as a piecewise linear approximation, formed from a set of sparse samples of $\text{lfs}(\mathbf{x})$. This sparse sampling is achieved by forming a coarse constrained Delaunay triangulation $\text{Del}(Y)$ of the points in the PLC $Y \in \mathcal{P}$. Additional vertices are inserted

to ensure that $\text{Del}(Y)$ is a conforming triangulation of the domain $\mathcal{P}$, such that no edge segments are encroached. The final supporting complex $\mathcal{S}_g$ for $\bar{h}_g(\mathbf{x})$ is constructed by performing a minimal refinement of $\text{Del}(Y)$. Using a modified version of Ruppert's Delaunay-refinement algorithm, any triangle $\tau \in \mathcal{S}_g$ is refined if it (i) *spans* the domain, having all three vertices $\mathbf{x}_i \in \tau$ on a boundary of $\mathcal{P}$, and (ii) has a sufficiently large radius-edge ratio, such that $\rho(\tau) \geq \bar{\rho}_g$. Since a minimal refinement is desired, a non-strict limit $\bar{\rho}_g$ is typically specified, and in this study, is set such that $\bar{\rho}_g = 4$. The resulting sparse complex $\mathcal{S}_g$ contains an approximation of the so-called *medial-axis* of the domain, wherein the additional points added during the refinement process are equidistant to their respective closest features in the domain $\mathcal{P}$. Importantly, it should be noted that since these points are positioned at the 'centre' of local geometrical features, they coincide with ridges and peaks in the local feature size function $\text{lfs}(\mathbf{x})$. Construction of the geometric size function $\bar{h}_g(\mathbf{x})$ for an example domain is illustrated in Figure 3.2.

A total mesh size function $\bar{h}(\mathbf{x})$ is finally obtained as a combination of the user-defined and geometric size functions, $\bar{h}_u(\mathbf{x})$ and $\bar{h}_g(\mathbf{x})$, such that $\bar{h}(\mathbf{x}) = \min(\bar{h}_u(\mathbf{x}), \bar{h}_g(\mathbf{x}))$ for all $\mathbf{x} \in |\mathcal{P}|$. A supporting complex $\mathcal{S} = \text{Del}(X_u \cup X_g)$ is built for $\bar{h}(\mathbf{x})$, where the vertices $X_u \in \mathcal{S}_u$ and $X_g \in \mathcal{S}_g$ are supports for the user-defined and geometric size functions, respectively. It should be noted that the total size function $\bar{h}(\mathbf{x})$ expresses the limiting size constraints imposed by both user-defined and geometric inputs.

3.3.2 Size Function Smoothing

The final stage in the construction of the mesh size function $\bar{h}(\mathbf{x})$ involves the imposition of a prescribed *smoothness* limit, g, such that, for any two points $\mathbf{x}_i, \mathbf{x}_j \in |\mathcal{P}|$, the local increase in size is bounded, with $\bar{h}(\mathbf{x}_j) \leq \bar{h}(\mathbf{x}_i) + g\,\|\mathbf{x}_i - \mathbf{x}_j\|$ and visa-versa. Such a function is said to be g-Lipschitz. Such limits on the smoothness of $\bar{h}(\mathbf{x})$ are introduced to ensure that the size constraints are consistent with desired element quality. Clearly, size functions that vary more slowly are expected to be congruent with improved element quality, while also leading to an increase in the number of elements in the output $|\mathcal{T}|$. The relationship between element shape quality and size function smoothness can be explored through a simple idealised model, considering an isosceles triangle with a base edge of length h and sides of length $(1+g)h$. Solving for g in terms of the small angle θ_{min} at the apex of the triangle gives $g = \frac{1}{2}(\sin(\frac{1}{2}\theta_{\text{min}}))^{-1} - 1$. Clearly, $g \to 0$ as $\theta_{\text{min}} \to 60.0°$. An element of moderate quality, with $\theta_{\text{min}} = 45.0°$, corresponds to $g \simeq 0.3$. It should be noted that this simple model is *optimistic*, with lower quality elements generated in practice due to a number of other factors.

I adapt the methods introduced by Persson and Strang [11, 12] to impose smoothness constraints on the size function $\bar{h}(\mathbf{x})$. This smoothing algorithm is summarised in Algorithm 3.3.1. Firstly, observe that a point $\mathbf{x}_j$ can only be smoothed from adjacent points $\mathbf{x}_i \in \mathcal{S}$ provided that $\bar{h}(\mathbf{x}_i) < \bar{h}(\mathbf{x}_j)$. This condition implies that an optimal ordering of points exists, for which the application of all smoothness constraints can be achieved in a single pass. An efficient greedy algorithm based on these observations first inserts all points $\mathbf{x} \in \mathcal{S}$ into a priority queue, sorted by the associated size function values $|\bar{h}(\mathbf{x})|$

Algorithm 3.3.1 g-Lipschitz Smoothing

```
1: function SmoothFunc(𝒮, h(x))
2:     Enqueue all vertices x ∈ 𝒮, Q ← h(x)
3:     while (Q ≠ ∅) do
4:         Dequeue vertex with lowest size, x_i ← Q
5:         for all (x_j adj. x_i in 𝒮) do          ▷ {limit pairwise gradients}
6:             Set h(x_j) = min(h(x_j), h(x_i) + g ‖x_i − x_j‖)
7:             Update priority queue, Q ← h(x_j)
8:         end for
9:     end while
10: end function
```

for each point. At each iteration of the main loop, the unprocessed point $\mathbf{x}_i$ of minimum current size value $\bar{h}(\mathbf{x}_i)$ is removed from the queue, and smoothness constraints are checked in its local neighbourhood. Specifically, all points $\mathbf{x}_j \in \mathcal{S}$ adjacent to $\mathbf{x}_i$ are examined, with their size values modified if they violate the local smoothness limit. This limit can be expressed via the linear inequality $\bar{h}(\mathbf{x}_j) \leq \bar{h}(\mathbf{x}_i) + g\,\|\mathbf{x}_i - \mathbf{x}_j\|$. The algorithm terminates when all points have been removed from the queue. The resulting size function $\bar{h}(\mathbf{x})$ ensures that for any two points $\mathbf{x}_i, \mathbf{x}_j$ connected by an edge $e \in \mathcal{S}$ the variation is bounded, such that $\bar{h}(\mathbf{x}_j) \leq \bar{h}(\mathbf{x}_i) + g\,\|\mathbf{x}_i - \mathbf{x}_j\|$ and visa-versa.

3.3.3 Discussion

While the size-optimal point-placement strategy presented in Section 3.2 is clearly reliant on the construction of a high-quality mesh size function, it is important to note that the overall Frontal-Delaunay algorithm is more flexible. Specifically, the Frontal-Delaunay algorithm does not explicitly require that the mesh size function $\bar{h}(\mathbf{x})$ correctly satisfy either (i) all geometric constraints imposed by the input domain $\mathcal{P}$, or (ii) an appropriate Lipschitz smoothness constraint. In fact, since the Frontal-Delaunay algorithm incorporates a standard Delaunay-refinement technique via the Type I and Type III point-placement strategies, setting $\bar{h}(\mathbf{x}) = \infty$ simply forces the algorithm to operate in pure Delaunay-refinement mode. Note that under such conditions, the satisfaction of all element shape and size constraints is still guaranteed – the algorithm simply operates without the benefit of the size-optimal point placement scheme. This robustness and flexibility is a key factor that differentiates the proposed Frontal-Delaunay method from conventional advancing-front techniques, that may sometimes fail, or generate low quality meshes when used in conjunction with an inappropriate mesh size function $\bar{h}(\mathbf{x})$.

3.4 Domains with Small Angles

The utility of both the Delaunay-refinement and Frontal-Delaunay algorithms presented in this chapter can be significantly improved by relaxing the geometric constraints on the domain $\mathcal{P}$, allowing PLC's containing sharp features to be meshed. Clearly, for PLC's containing very acute angles, it is not possible to ensure that all triangles in the resulting mesh $\mathcal{T} = \text{Del}(X)$ satisfy the expected angle bounds whilst concurrently enforcing domain conformity. For example, it is impossible to eliminate the small angle at the apex of a

needle shaped domain, irrespective of both the number and position of the Steiner vertices inserted. Non-convergence may ensue if the unmodified Delaunay-refinement or Frontal-Delaunay algorithms are used to mesh domains containing sufficiently sharp features, with the Steiner vertices positioned on edges adjacent to small angles potentially leading to an infinite cascade of *mutual-encroachment*. When dealing with domains subtending small angles it is instead necessary to accept that some sacrifice of element quality is necessary in the vicinity of sharp geometric features and that a set of special cases must be identified to ensure that convergence is achieved.

A number of techniques have been developed to extend Ruppert's Delaunay-refinement algorithm to handle general domains that incorporate arbitrarily small acute angles. Specifically, both the *concentric shell* method, originally proposed by Ruppert in [17] and the so-called *seditious edge* scheme introduced by Miller, Pav and Walkington in [9] introduce a set of special case rules and thresholds for the refinement of elements in the neighourhood of a sharp feature. In both these methods, convergence is guaranteed by ensuring that a subset of low-quality elements immediately adjacent to any sharp features in $\mathcal{P}$ are considered to be *un-refinable*, and are allowed to persist through the refinement process. In the present study, I pursue a slightly different approach, and, as such, refer the reader to the comprehensive treatment of these topics presented in [2, Chapter 6] for additional information.

3.4.1 Protecting Disks

In the present study, I adopt a strategy inspired by the idea of *protective-collars*, as introduced by Rand and Walkington [13, 14], in which a small subset of the domain, adjacent to any sufficiently sharp features in $\mathcal{P}$ is pre-processed and *quarantined* from subsequent refinement. Such a formulation is similar to the so-called *protecting-balls* technique of Cheng, Dey and Ramos [1] for higher-dimensional problems. Given a general PLC $\mathcal{P}$, the *protection* process seeks to enclose any sharp features present in the domain inside a set of Euclidean balls, within which refinement, through the introduction of new Steiner vertices, is prohibited. I adopt a similar procedure in this study, in which a set of *protecting-disks* $\mathcal{D}$ are placed at the apex of any sharp features in the domain $\mathcal{P}$. The disks $\mathcal{D}$ define a *protected region* within which refinement is disallowed. The radii of the disks is selected to ensure that the protected regions are sufficiently small with respect to local constraints.

Proposition 3.6 (locality)**.** *Let the radius r_i of a protecting disk $d_i \in \mathcal{D}$ be sufficiently small, such that $r_i \leq \left(\frac{1}{3}\right) \mathrm{lfs}(\mathbf{x}_a)$, where $\mathbf{x}_a \in \mathcal{P}$ is the vertex positioned at the apex of the associated sharp feature. The disk d_i is well-separated from all disjoint features in the domain, including all vertices $\mathbf{x} \in \mathcal{P}$, $\mathbf{x} \neq \mathbf{x}_a$, edges $e \in \mathcal{P}$, $e \cap \mathbf{x}_a = \emptyset$ and disks $d_j \in \mathcal{D}$, $d_j \neq d_i$.*

Proof. Let d_i be a disk of radius $r_i \leq \gamma \, \mathrm{lfs}(\mathbf{x}_i)$ centred on $\mathbf{x}_i$. Recalling that $\mathrm{lfs}(\mathbf{x}_i)$ is the minimum distance from the vertex $\mathbf{x}_i$ to any disjoint feature in the domain $\mathcal{P}$, it is clear that the minimum distance between any point in the disk d_i and a disjoint feature in $\mathcal{P}$ is at least r_i when $\gamma \leq \frac{1}{2}$. Furthermore, the minimum distance between any two disks

$d_i, d_j \in \mathcal{D}$ is at least r_i when $\gamma \leq \frac{1}{3}$, even when such disks are centred on opposite ends of the same edge $e_i \in \mathcal{P}$. □

In practice, Proposition 3.6 is implemented by setting $r_i = \bar{h}(\mathbf{x}_i)$ for each disk, and enforcing local modifications to the mesh size function, such that $\bar{h}(\mathbf{x}_i) \leq \left(\frac{1}{3}\right) \text{lfs}(\mathbf{x}_i)$ in the neighbourhood of any sharp features in the PLC. Such a strategy ensures a close coupling between the size of the protected regions and the magnitude of the local mesh size function, and has been found to produce smoothly graded meshes in practice. This approach also ensures that sharp features are resolved at an appropriate scale when $\bar{h}(\mathbf{x_i}) \leq \text{lfs}(\mathbf{x}_i)$.

Given a set of protecting disks $\mathcal{D}$, a set of new Steiner vertices is first introduced at intersections between the disks $d_i \in \mathcal{D}$ and the input domain $\mathcal{P}$. By placing a 'ring' of so-called *collar-vertices* at a constant radius r_i about each apex $\mathbf{x}_a$, a conforming Delaunay triangulation of the protected region of $\mathcal{P}$ is induced.

Proposition 3.7 (conformity)**.** *Let the set of collar vertices X_s include all intersections $\mathcal{D} \cap E \neq \emptyset$, where $\mathcal{D}$ is the set of protecting disks, and $E \subseteq \mathcal{P}$ is the set of edges in the* PLC*. The Delaunay tessellation* $\text{Del}(X \cup X_s)$ *is a conforming triangulation of the protected regions in $\mathcal{P}$.*

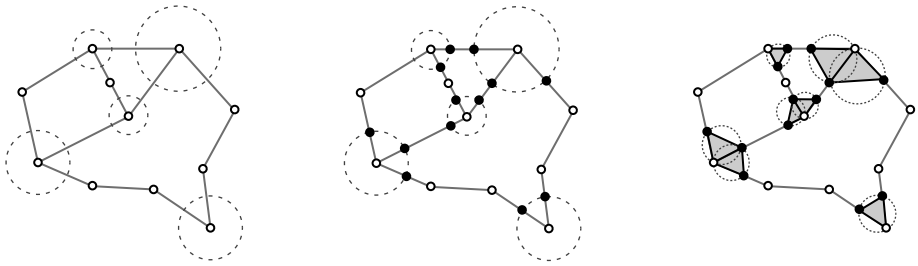
Proof. Considering any given sharp feature in the domain $\mathcal{P}$, a pair of vertices $\mathbf{x}_i, \mathbf{x}_j \in X_\text{s}$ are positioned at the intersections $d_i \cap e_i$ and $d_i \cap e_j$, where $d_i \in \mathcal{D}$ is the associated protecting disk and $e_i, e_j \in \mathcal{P}$ are the edges associated with the sharp feature. Noting that the vertices $\mathbf{x}_i, \mathbf{x}_j$ are equidistant from the apex $\mathbf{x}_\text{a}$, the triplet $[\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_\text{a}]$ defines an isosceles triangle σ_a that *conforms* to local edge constraints. The circumcentre $\mathbf{c}_\sigma$ clearly lies within σ, positioned along the vector $[\mathbf{x}_m, \mathbf{x}_\text{a}]$, where $\mathbf{x}_m$ is the midpoint of the edge $[\mathbf{x}_i, \mathbf{x}_j]$. The associated circumradius r_σ is bounded as a result, such that $r_\sigma \leq r_i$, where r_i is the radius of the protecting disk d_i. Using Proposition 3.6, the disk d_i is known to be empty of any non-adjacent vertices $\mathbf{x}_k \in \mathcal{P}$. Additionally, note that any adjacent collar vertices $\mathbf{x}_l \in X_\text{s}$ are positioned on the circumference of a common ball of radius r_i. The circumball of σ is empty as a consequence, and $\sigma \in \text{Del}(X \cup X_\text{s})$ by the Delaunay criterion. □

Following this initialisation about sharp features, the meshing problem is reduced to the task of triangulating the remaining *protected* domain $\mathcal{P}' = \mathcal{P} \setminus \mathcal{T}_\mathcal{D}$, where $\mathcal{T}_\mathcal{D}$ is the set of all collar simplexes created during the initial protection phase. Importantly, the protected domain is known to be free of any sharp features.

Proposition 3.8 (local convexity)**.** *Let $\mathcal{T}_\mathcal{D}$ be the conforming triangulation of the protected region of a general* PLC *$\mathcal{P}$. The protected domain $\mathcal{P}' = \mathcal{P} \setminus \mathcal{T}_\mathcal{D}$ does not contain sharp features, with all edges $E \subseteq \mathcal{P}'$ meeting at non-acute angles.*

Proof. Considering any given sharp feature in the domain $\mathcal{P}$, recall that an isosceles triangle $\sigma = [\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_\text{a}]$, where $\mathbf{x}_i, \mathbf{x}_j$ are the associated collar vertices, is introduced to *protect* the sharp feature centred at $\mathbf{x}_\text{a}$. Let the angle of the feature be α and the

Figure 3.3: Illustration of the protection strategy for a general PLC, showing (i) the set of protecting disks $\mathcal{D} = \{d_1, \ldots, d_n\}$ associated with sharp features in the domain $\mathcal{P}$, (ii) the set of *collar-vertices* X_s introduced at the intersections $\mathcal{D} \cap E \neq \emptyset$, and (iii) the set of *collar-simplexes* $\mathcal{T}_\mathcal{D}$ and their associated circumdisks $\mathcal{D}_\tau$.



(i) (ii) (iii)

associated angles induced in the protected domain $\mathcal{P}'$ be β. Considering that the protected edge $[\mathbf{x}_i, \mathbf{x}_j]$ is the base of an isosceles triangle, it is clear that the included angle $\beta = \frac{1}{2}(180° + \alpha) \geq 90°$. Recalling that all features in $\mathcal{P}$ for which $\alpha \leq 90°$ are protected, it is clear that edges in the protected domain $\mathcal{P}'$ meet at non-acute angles. □

When triangulating the protected domain $\mathcal{P}'$, any *collar-simplexes* – that is, any elements $\tau_i \in \mathcal{T}_\mathcal{D}$ – are preserved throughout the subsequent refinement process. The Delaunay-refinement and Frontal-Delaunay methods presented in Algorithm 3.1.1 are modified to additionally test the suitability of each candidate Steiner vertex, declining to perform refinements that would result in the introduction of new vertices within the circumball of a collar simplex. The set of protected circumballs is denoted $\mathcal{D}_\tau$. Such filtering is accomplished efficiently by storing the collection $\mathcal{D}_\tau$ in a supporting AABB-tree. A subset of low-quality elements in the local neighbourhood of any sharp features in $\mathcal{P}$ are preserved as a result, and these elements appear in the final triangulation $\mathcal{T} = \text{Del}(X)$, in violation of the desired shape constraints $\bar{\rho}$. I refer to these modified Delaunay-refinement and Frontal-Delaunay algorithms as the *protected* variants.

Proposition 3.9 (performance)**.** *Let $\mathcal{T} = \text{Del}(X)$ be a triangulation generated by the protected Delaunay-refinement or Frontal-Delaunay algorithms for a domain $\mathcal{P}$. The tessellation $\mathcal{T}$ is a conforming triangulation of $\mathcal{P}$, and, for all elements in the sub-mesh $\mathcal{T}_{\mathcal{P}'} = \{\tau \in \mathcal{T} \,|\, \mathbf{c}_\tau \cap \mathcal{D}_\tau = \emptyset\}$, local shape- and size-based constraints are satisfied, such that $\rho(\tau) \leq \bar{\rho}$ and $h(\tau) \leq \bar{h}(\mathbf{x}_\tau)$, where $\mathbf{c}_\tau$ are the element circumcentres, $\bar{\rho}$ is a prescribed radius-edge ratio, and $\bar{h}(\mathbf{x}_\tau)$ is a user-defined mesh size function sampled at the element circumcentres.*

Proof. Recalling Propositions 3.3 and 3.1, and noting, by Proposition 3.8, that the protected domain $\mathcal{P}'$ includes only non-acute features, the (unprotected) Delaunay-refinement and Frontal-Delaunay algorithms (3.1.1) are guaranteed to produce a conforming triangulation $\mathcal{T}_{\mathcal{P}'}$ of the protected domain $\mathcal{P}'$, satisfying all size- and shape-based constraints. By rejecting refinement operations that would lead to the introduction of new Steiner vertices lying inside any circumdisk $d_i \in \mathcal{D}_\tau$ associated with a protected element $\tau_i \in \mathcal{T}_\mathcal{D}$, the set of collar simplexes $\mathcal{T}_\mathcal{D}$ is preserved throughout the refinement process, ensuring

that the final tessellation $\mathcal{T} = \mathcal{T}_{\mathcal{P}'} \cup \mathcal{T}_{\mathcal{D}}$ is a conforming Delaunay triangulation of the original domain $\mathcal{P}$. □

3.5 Results & Discussion

The performance of the planar meshing algorithms introduced in this chapter was investigated experimentally, with both the Delaunay-refinement and Frontal-Delaunay algorithms developed in Sections 3.1 and 3.2 used to mesh a series of eleven benchmark problems of varying size and complexity. A range of test domains were chosen from a diverse set of application areas, including problems from geospatial processing, computational fluid dynamics, and solution adaptive numerical simulation. Meshes for each domain, generated using the Frontal-Delaunay approach, are shown in Figure 3.4, except for the adaptive meshing examples, which are deferred until Figure 3.8. Note that a number of the test-cases include small acute angles. Both the Frontal-Delaunay algorithm described in Section 3.2 and Ruppert's Delaunay-refinement algorithm described in Section 3.1 were implemented, allowing the performance and output of the two algorithms to be compared side-by-side. Due to the similarities in structure between the two algorithms, a common code-base was used, with the algorithms differing only in the type of Steiner vertices inserted and in the manner in which the queue of bad triangles is updated, as discussed in Section 3.2. Both algorithms are built using the `TRIPOD` and `LUMBERJACK` packages – making use of the efficient incremental Delaunay triangulation and spatial indexing frameworks presented in Chapter 2. The Frontal-Delaunay and Delaunay-refinement algorithms are included in the `JIGSAW` package – a new mesh generation library built using the algorithms developed in this thesis. Both algorithms are implemented in C++ and compiled as 64-bit executables.

Meshes generated using the new Frontal-Delaunay algorithm for the full set of benchmarks are presented in Figure 3.4, demonstrating the effectiveness of the new strategy in practice. These meshes are generated using relatively coarse element size constraints, with the mesh size functions $\bar{h}(\mathbf{x})$ constructed using a Lipschitz smoothness parameter $g = 3/10$. Tight bounds on element shape-quality are imposed, setting $\bar{\rho}$ to achieve $\theta_{\min} \simeq 29^\circ$ and $\theta_{\max} \simeq 122^\circ$. Qualitatively, these results confirm that the new Frontal-Delaunay algorithm is successful in practice, generating high-quality conforming meshes for a series of complex inputs. It can also be seen that smoothly graded meshes are generated in all cases, with high resolution regions adjacent to small geometrical features transitioning to lower resolution areas in the 'interior' of the domains. It is also noted that convergence is achieved in all cases, despite the element angle bound $\theta_{\min} \simeq 29^\circ$ significantly exceeding the theoretical limit $\theta_{\min} \simeq 20.7^\circ$ at which convergence is guaranteed. This behaviour is consistent with the results of a number of previous studies, including, for example, investigations by Shewchuk [18, 19] and Üngör [6, 20] who show that Delaunay-refinement algorithms typically outperform their theoretical bounds in practice. Shewchuk remarks that convergence is typically observed for $\theta_{\min} \simeq 33.8^\circ$, although rarely higher. Similar behaviour has been observed throughout this study, confirming that the size-optimal point-placement strategy used in the Frontal-Delaunay

Figure 3.4: Benchmark problems for planar meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29°$. Normalised histograms of element area-length ratio $a(\tau)$, plane angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ are also shown.

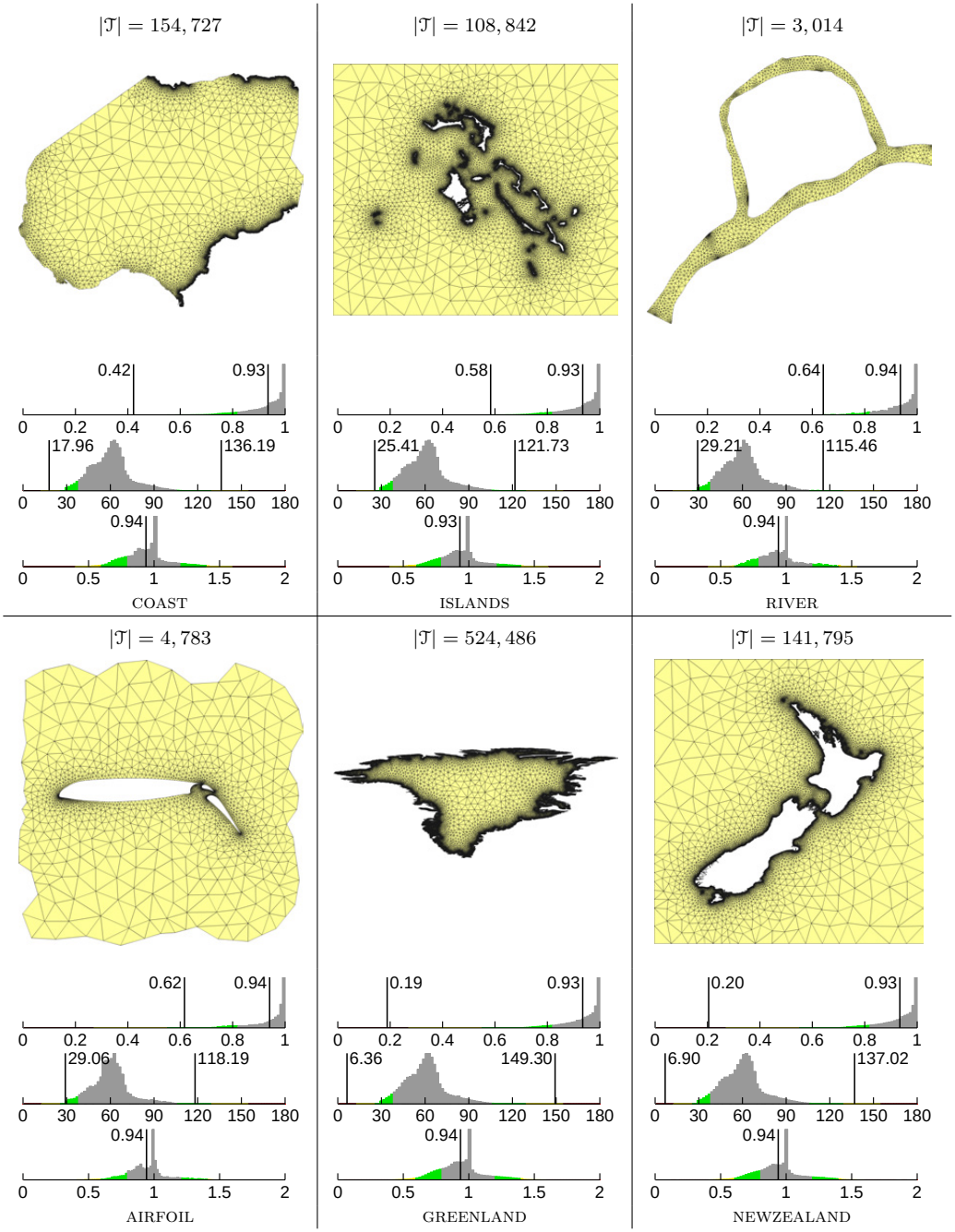
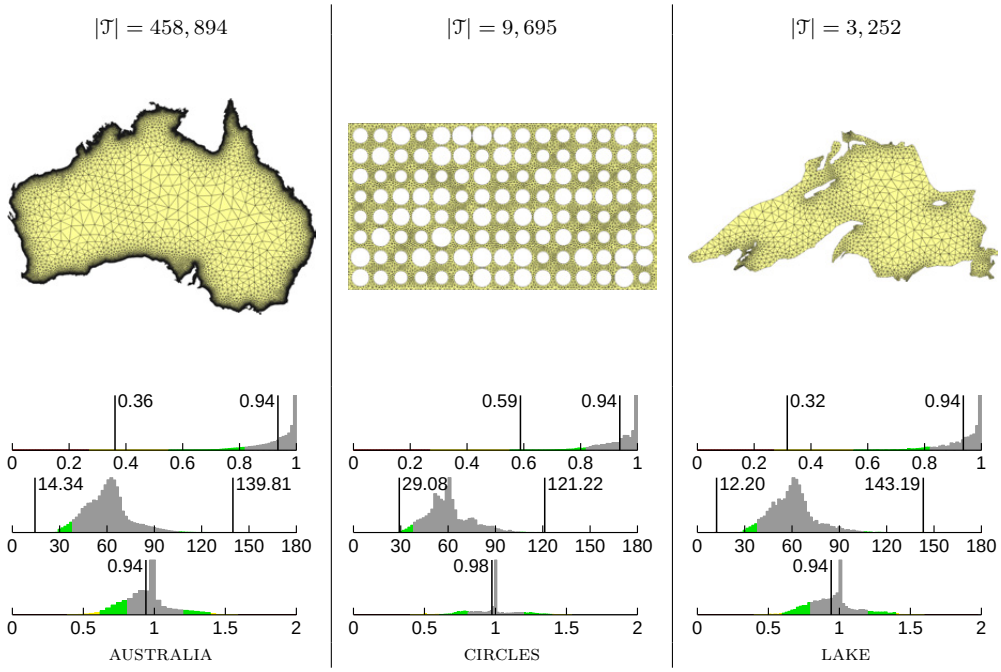


Figure 3.5: Benchmark problems for planar meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29°$. Normalised histograms of element area-length ratio $a(\tau)$, plane angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ are also shown.



algorithm does not have a negative effect on practical performance. Additionally, it is also important to note that convergence is achieved for a number of inputs containing input angles as small as 6.4°, confirming the effectiveness of the *protecting-disks* strategy used for domains containing sharp features.

3.5.1 Comparative Performance

The results of a comparative performance study, contrasting the effectiveness of the Frontal-Delaunay and Delaunay-refinement meshing algorithms, is presented in Figures 3.6 and 3.7 and includes detailed results for the AIRFOIL and LAKE test problems. Specifically, the effectiveness of the new size-driven refinement scheme is addressed, comparing a range of meshes generated using the Frontal-Delaunay and Delaunay-refinement algorithms for a range of different input parameters. The resulting meshes are compared in terms of their size- and shape-quality, and their underlying structure. In addition to the detailed results presented for the AIRFOIL and LAKE problems, a simplified set of comparisons are also tabulated for the full set of benchmark problems.

3.5.1.1 Size-driven Refinement

A detailed study of meshes generated for the AIRFOIL and LAKE problems, presented in Figures 3.6 and 3.7, examines the impact of the size-optimal Type II point-placement strategy introduced in Section 3.2. A set of meshes were generated for both test cases using low, medium and high resolution settings, where the associated mesh size functions

$\bar{h}(\mathbf{x})$ were constructed using Lipschitz smoothness values of $g = {}^{3}/_{10}$, $g = {}^{2}/_{10}$ and $g = {}^{1}/_{10}$, respectively. The radius-edge threshold was held constant across all test cases, such that $\theta_{\min} \simeq 29°$. For all test problems, element quality has been catalogued, with normalised histograms of element area-length $a(\tau)$, plane angle $\theta(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean area-length measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length. Note that the domain for the LAKE test case includes a small acute angle, seen as an outlier in distributions of $a(\tau)$ and $\theta(\tau)$.

An analysis of Figures 3.6 and 3.7 shows, firstly, that both the Frontal-Delaunay and Delaunay-refinement algorithms generate high-quality meshes for all test cases – satisfying the required element angle thresholds, except in regions adjacent to sharp features. Inspection of distributions of $a(\tau)$, $\theta(\tau)$ and $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ show that the Frontal-Delaunay algorithm consistently outperforms the Delaunay-refinement scheme, generating meshes with higher mean area-length ratios and 'tight' distributions of element plane angle and relative edge length in all cases. The most significant difference in behaviour between the two algorithms appears to be in the way that mesh size constraints are imposed. Given the narrow distributions of relative edge length, strongly clustered about $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$, it is clear that the Frontal-Delaunay algorithm successfully generates meshes that accurately conform to the imposed mesh size constraints. In contrast, output generated using the Delaunay-refinement scheme is seen to incorporate significant sizing error, typified by 'broad' distributions of relative edge length straddling $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$. While the mean relative edge lengths are comparable in all cases, the Frontal-Delaunay algorithm clearly generates much more accurate output on an element-by-element basis.

Additionally, it is evident that the quality of meshes generated using the Frontal-Delaunay algorithm improves as $g \to 0$, as indicated by the increase in mean $a(\tau)$, the narrowing of the $\theta(\tau)$ distribution about 60.0°[1] and the narrowing of the distribution of relative edge length about $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$. In contrast, similar analysis shows that the Delaunay-refinement results are essentially independent of the magnitude of the mesh size function, $h(\mathbf{x})$, with distributions of $a(\tau)$, $\theta(\tau)$ and $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ seen to be broadly consistent across all cases. Visually, the enhanced quality of the meshes generated using the Frontal-Delaunay algorithm is evident, with a marked increase in mesh smoothness and sub-structure obvious. Meshes generated by both algorithms are seen to be similar size, with neither algorithm showing a significant or consistent deviation in element count $|\mathcal{T}|$.

An analysis of the Steiner refinement strategies logged throughout this study show that the majority of the off-centres chosen by the Frontal-Delaunay algorithm were either size-optimal Type II points ($\simeq 60\%$) or conventional Type I circumcentres ($\simeq 40\%$). The bias toward size-optimal off-centres was also observed to increase as $\bar{h}(\mathbf{x}) \to 0$. The use of shape-optimal Type III off-centres was noted to be rare ($\ll 1\%$). These results are not surprising, simply indicating that when the magnitude of the mesh size function $\bar{h}(\mathbf{x})$ is sufficiently small, the element size constraints dominate and the Type II point-placement

[1] Triangles with ideal shape quality are equilateral, explaining why $\theta(\tau) \to 60.0°$ as shape-quality is improved.

Figure 3.6: Size-driven refinement study for the AIRFOIL problem, showing meshes generated using the Frontal-Delaunay (upper) and Delaunay-Refinement (lower) algorithms. Meshes are generated with increasing values of size function smoothness, $g = {}^{3}/_{10}$, $g = {}^{2}/_{10}$ and $g = {}^{1}/_{10}$ from left to right. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29°$. Element counts $|\mathcal{T}|$ are included for each case. Normalised histograms of element area-length ratio $a(\tau)$, plane angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ are also shown.

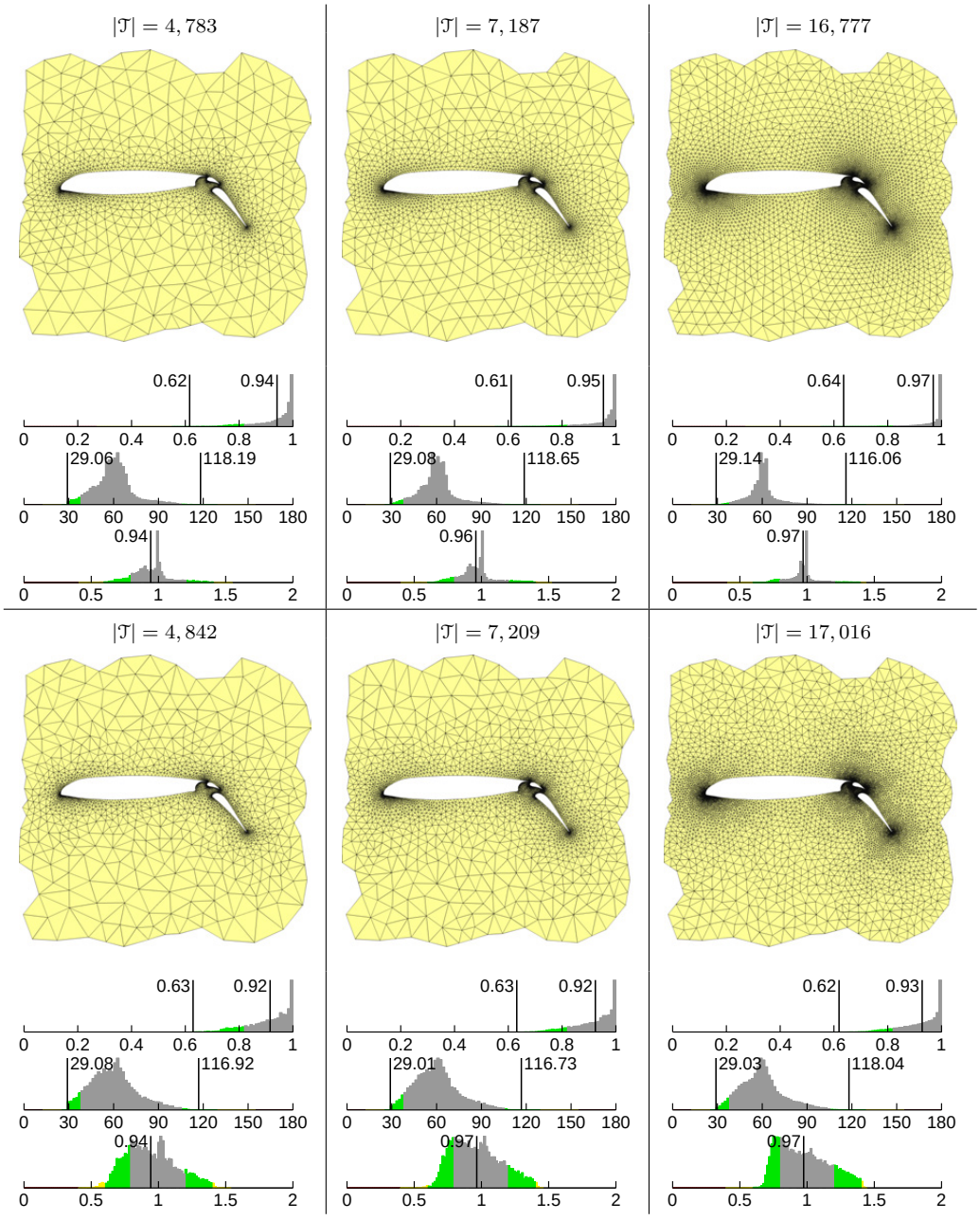


Figure 3.7: Size-driven refinement study for the LAKE problem, showing meshes generated using the Frontal-Delaunay (upper) and Delaunay-Refinement (lower) algorithms. Meshes are generated with increasing values of size function smoothness, $g = {}^{3}/_{10}$, $g = {}^{2}/_{10}$ and $g = {}^{1}/_{10}$ from left to right. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29°$. Element counts $|\mathcal{T}|$ are included for each case. Normalised histograms of element area-length ratio $a(\tau)$, plane angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ are also shown.

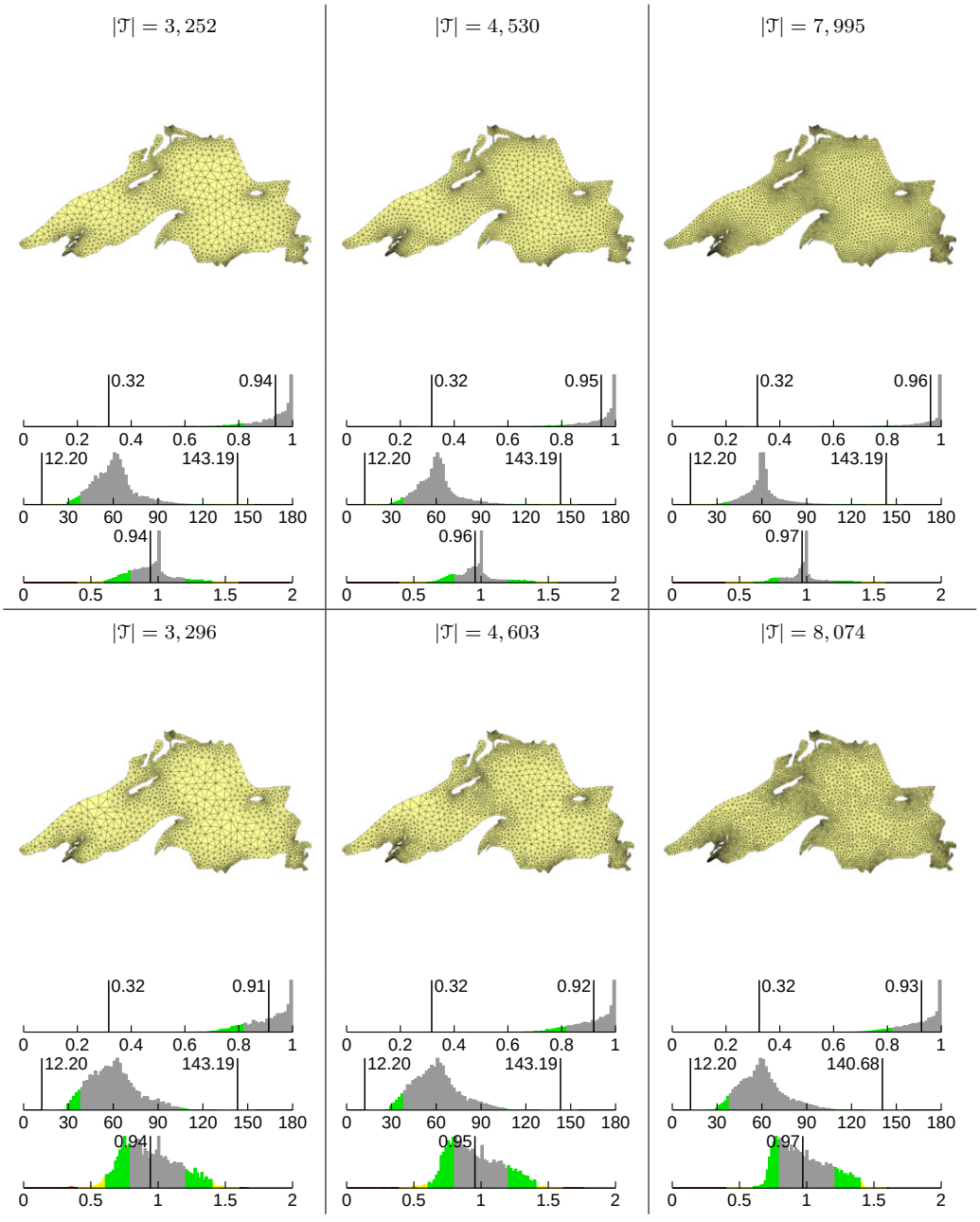


Table 3.1: Planar meshing study, comparing the performance of the Frontal-Delaunay and Delaunay-refinement algorithms on the full set of benchmark problems. Meshes were generated using medium resolution settings, such that the mesh size functions $\bar{h}(\mathbf{x})$ are constructed with $g = 2/10$. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29^\circ$. Results listed include: total element count $|\mathcal{T}|$, total runtime $t(s)$, mean and minimum area-length ratios $\overline{a(\tau)}$, $a(\tau)_{\min}$ and plane angle bounds $\theta(\tau)_{\min}$, $\theta(\tau)_{\max}$.

Domain	Frontal-Delaunay						Delaunay-refinement					
	$\lvert\mathcal{T}\rvert$	$t(s)$	$a(\tau)_{\min}$	$\overline{a(\tau)}$	$\theta(\tau)_{\min}$	$\theta(\tau)_{\max}$	$\lvert\mathcal{T}\rvert$	$t(s)$	$a(\tau)_{\min}$	$\overline{a(\tau)}$	$\theta(\tau)_{\min}$	$\theta(\tau)_{\max}$
AIRFOIL	7,187	0.09	0.61	0.95	29.1°	118.7°	7,209	0.06	0.63	0.92	29.0°	116.7°
LAKE	4,530	0.04	0.32	0.95	12.2°	143.2°	4,603	0.04	0.32	0.92	12.2°	143.2°
CIRCLES	10,131	0.08	0.62	0.94	29.1°	117.6°	10,231	0.08	0.62	0.91	29.0°	118.2°
RIVER	4,218	0.04	0.61	0.95	29.2°	119.5°	4,306	0.04	0.61	0.92	29.1°	119.3°
NEWZEALAND	212,543	1.44	0.20	0.95	6.9°	141.4°	214,965	1.38	0.20	0.92	6.9°	141.4°
ISLANDS	166,293	1.05	0.59	0.95	25.4°	121.1°	168,212	0.96	0.59	0.92	25.4°	121.2°
COAST	229,147	1.53	0.42	0.95	18.0°	136.2°	231,559	1.44	0.42	0.92	18.0°	136.2°
AUSTRALIA	689,868	5.23	0.36	0.95	14.3°	139.8°	697,730	5.25	0.36	0.92	14.3°	139.8°
GREENLAND	710,766	5.72	0.19	0.95	6.4°	148.9°	789,969	6.16	0.19	0.92	6.4°	148.9°
CAVITY	34,572	0.22	0.64	0.98	29.6°	115.6°	36,331	0.19	0.60	0.93	29.0°	119.6°
CYLINDER	67,618	0.45	0.65	0.98	29.0°	114.3°	72,421	0.41	0.59	0.93	29.0°	120.6°

scheme is preferred.

3.5.1.2 Overall Comparisons

In addition to the detailed comparisons presented previously, a simplified set of comparative results were also obtained for all benchmark problems, contrasting the performance of the Frontal-Delaunay and Delaunay-refinement approaches. The results of this study are presented in Table 3.1. Meshes were generated using medium resolution settings, with the mesh size function $\bar{h}(\mathbf{x})$ constructed with $g = 2/10$. Again, tight bounds on element radius-edge ratios were specified, such that $\theta_{\min} \simeq 29^\circ$ and $\theta_{\max} \simeq 122^\circ$. Analysis of the results in Table 3.1 confirm the trends observed in the detailed analysis carried out for the AIRFOIL and LAKE problems – that, given an appropriate mesh size function $\bar{h}(\mathbf{x})$, the proposed Frontal-Delaunay algorithm produces meshes that are of significantly higher shape- and size-quality than those generated using the conventional Delaunay-refinement algorithm. Total run-times for the algorithms are also tabulated and show, firstly, that the implementations developed in this study are efficient, generating meshes containing $100,000$'s of elements in a matter of seconds. These results also show that the new off-centre point-placement scheme used in the Frontal-Delaunay algorithm imposes a small additional cost, with the total runtime increased by 15–20%. This increase in computational work is associated with the local iteration used when positioning the size-optimal Type II Steiner vertices.

3.5.2 Solution Adaptive Meshing

The performance of the planar meshing algorithms for problems involving user specified mesh size functions $\bar{h}_u(\mathbf{x})$ was also assessed. In Figure 3.8, the results of an adaptive meshing study are presented, in which a sequence of graded meshes are generated for a pair of example problems from computational fluid dynamics, including the driven flow in a square cavity and the flow over a pair of cylindrical cross-sections. Numerical simulations were performed using a Navier-Stokes solver previously developed by the author in [5], which is based on a vertex-centred finite-volume framework. An iterative solution-adaptive re-meshing process was investigated, in which the numerical solution obtained on a given mesh $\mathcal{T}^k$ used to generate a new mesh size function $\bar{h}_u(\mathbf{x})^{k+1}$. A new mesh $\mathcal{T}^{k+1}$ was then generated to conform to $\bar{h}_u(\mathbf{x})^{k+1}$, and the process repeated. Initial meshes were generated based on geometric information alone, with $\bar{h}_u(\mathbf{x})^{k=0} = \infty$. At each iteration, the meshes were adapted using a simple measure of the flow characteristics, with the new size function based on the magnitude of the local *vorticity* of the flow, $\omega(\mathbf{x}) = \nabla \times \mathbf{u}(\mathbf{x})$, such that $\bar{h}_u(\mathbf{x}) = \alpha/|\omega(\mathbf{x})|$, where $\alpha > 0$ is a scaling constant. The meshes generated using both the Frontal-Delaunay and Delaunay-refinement algorithms after 3 adaptive meshing cycles are presented in Figure 3.8. In all cases, the radius-edge ratio limit was set such that $\theta_{\min} = 29^\circ$ and the mesh size function smoothed using $g = 2/10$. For all test problems, element quality is catalogued, with normalised histograms of the element area-length $a(\tau)$, plane angle $\theta(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. The histograms further highlight the minimum and mean area-length

measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length.

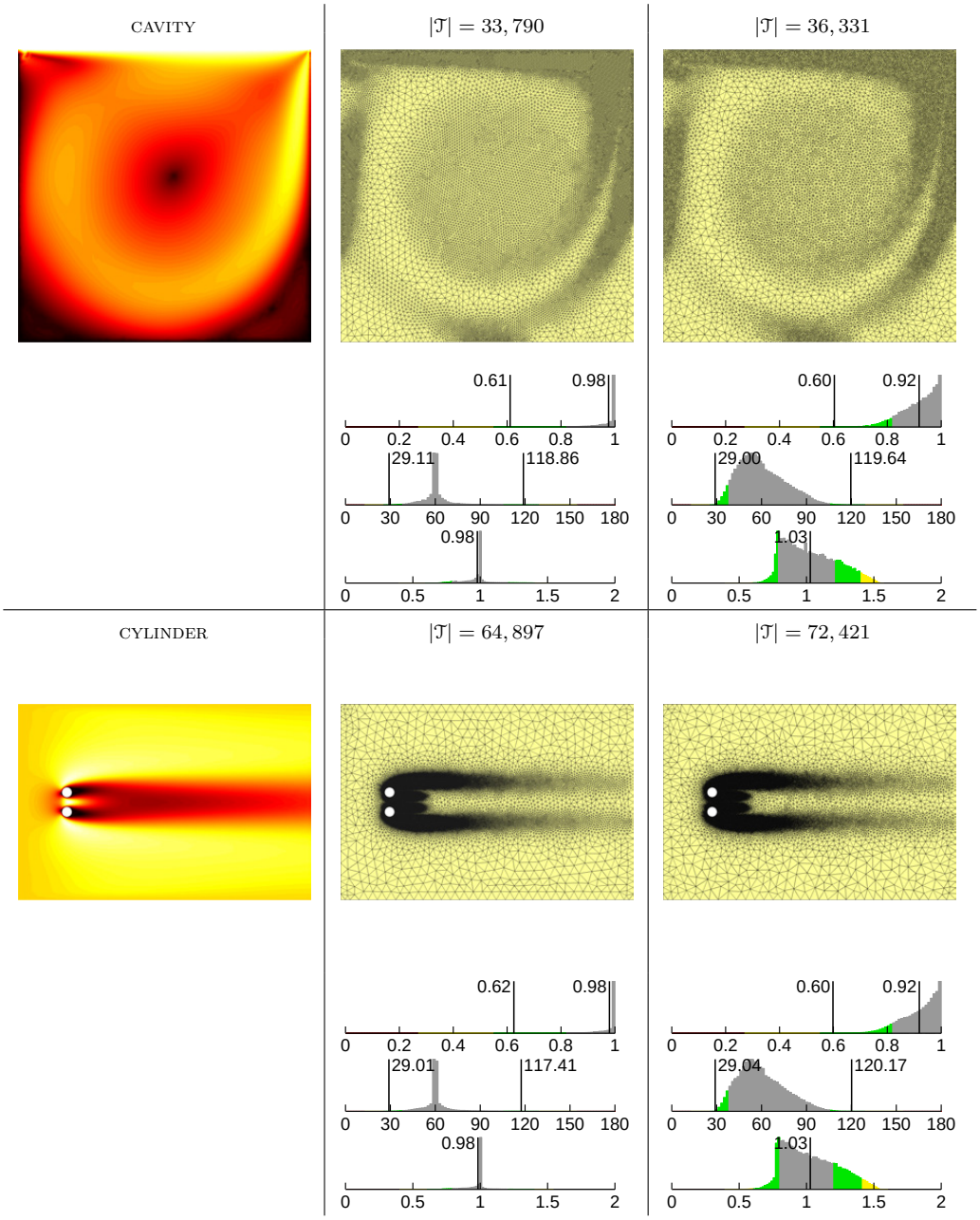
Analysis of these adaptive results confirm many of the trends discussed previously, with the Frontal-Delaunay algorithm generating meshes of significantly higher quality when compared to the conventional Delaunay-refinement method. Specifically, it is noted that distributions of both element quality $a(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are improved significantly when using the Frontal-Delaunay method, with the vast majority of elements consisting of near-perfect shape-quality and size. It is also observed that the Frontal-Delaunay algorithm produces meshes of slightly smaller size. A combination of improved element quality and reduced mesh size typically leads to improvements in efficiency in the associated numerical simulation. I do not seek to explicitly quantify this effect in this study, although it is an interesting avenue for future research. Overall, these results demonstrate the effectiveness of the proposed Frontal-Delaunay method when generating high-quality adaptive meshes for numerical simulation.

3.6 Conclusions

In this chapter, I have presented a pair of algorithms designed for high-quality mesh generation in the plane. The first algorithm is a conventional Delaunay-refinement scheme due to Ruppert [17], in which the circumcentres of poor quality triangles are inserted into the mesh as new Steiner vertices. In the second algorithm, I have developed a new Frontal-Delaunay technique, building on ideas previously introduced by Üngör [20] and Rebay [15], in which generalised Steiner vertices are inserted along edges in the associated Voronoi diagram. This new approach allows for the insertion of both size- and shape-optimal Steiner points, leading to a hybrid strategy that combines many of the advantages of advancing-front and Delaunay-refinement techniques. A series of comparative experimental studies confirm the effectiveness of this new technique, demonstrating that significant improvements in element quality are typically achieved in practice. Importantly, it has also been demonstrated that the new Frontal-Delaunay algorithm achieves much of the same theoretical optimality as conventional Delaunay-refinement schemes, satisfying constraints on element radius-edge ratios and edge length. Results show that the new algorithm is an effective hybridisation of existing mesh generation techniques, combining the high element quality and mesh structure of advancing-front techniques with the theoretical guarantees of Delaunay-refinement schemes.

A number of avenues exist for future investigations – improving or generalising the Delaunay-refinement and Frontal-Delaunay meshing algorithms presented in this study. Specifically, further generalisations of the off-centre techniques could be investigated, following, for example, the strategies outlined by Erten and Üngör in [6], in which Steiner vertices are inserted along *non-adjacent* segments in the associated Voronoi diagram. In the context of shape optimality alone, such methods have been reported to improve the performance of a planar meshing algorithm, further reducing the output size and improving the element angle distribution. Initial analysis suggests that these methods could also be formulated in terms of size-optimal constraints, allowing them to be incorporated into the Frontal-Delaunay framework introduced in this study.

Figure 3.8: Adaptive meshing study for the CAVITY and CYLINDER test cases, showing the numerical solutions and associated meshes after 3 iterations of solution-adaptive meshing. Contours of velocity magnitude are shown (left). Meshes generated using the Frontal-Delaunay (centre) and Delaunay-refinement (right) algorithms are adapted to resolve high shear regions in the flow. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29^\circ$. Normalised histograms of element area-length ratio $a(\tau)$, plane angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ are also shown.



References

[1] Cheng, S. W., Dey, T. K., and Ramos, E. A. Delaunay Refinement for Piecewise Smooth Complexes. *Discrete & Computational Geometry 43*, 1 (2010), 121–166.

[2] Cheng, S. W., Dey, T. K., and Shewchuk, J. R. *Delauay Mesh Generation.* Taylor & Francis, New York, 2013.

[3] Chernikov, A. N., and Chrisochoides, N. P. Generalized insertion region guides for Delaunay mesh refinement. *SIAM Journal on Scientific Computing 34*, 3 (2012), A1333–A1350.

[4] Chew, L. P. Guaranteed-quality Triangular Meshes. Tech. rep., Cornell University, Department of Computer Science, Ithaca, New York, 1989.

[5] Engwirda, D. Unstructed Mesh Methods for the Navier-Stokes Equations. Undergraduate honors thesis, Sydney, NSW., 2005.

[6] Erten, H., and Üngör, A. Quality Triangulations with Locally Optimal Steiner Points. *SIAM J. Sci. Comp. 31*, 3 (2009), 2103–2130.

[7] Foteinos, P. A., Chernikov, A. N., and Chrisochoides, N. P. Fully generalized two-dimensional constrained Delaunay mesh refinement. *SIAM Journal on Scientific Computing 32*, 5 (2010), 2659–2686.

[8] Mavriplis, D. J. An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *Journal of Computational Physics 117*, 1 (1995), 90 – 101.

[9] Miller, G. L., Pav, S. E., and Walkington, N. J. When and Why Delaunay Refinement Algorithms Work. *International Journal of Computational Geometry & Applications 15*, 01 (2005), 25–54.

[10] Müller, J.-D., Roe, P. L., and Deconinck, H. A frontal approach for internal node generation in Delaunay triangulations. *International Journal for Numerical Methods in Fluids 17*, 3 (1993), 241–255.

[11] Persson, P. O. *Mesh Generation for Implicit Geometries.* PhD thesis, Boston, Massachusetts, 2004.

[12] Persson, P.-O. Mesh Size Functions for Implicit Geometries and PDE-based Gradient Limiting. *Engineering with Computers 22*, 2 (2006), 95–109.

[13] Rand, A., and Walkington, N. 3d Delaunay refinement of sharp domains without a local feature size oracle. In *Proceedings of the 17th International Meshing Roundtable.* Springer, 2008, pp. 37–54.

[14] Rand, A., and Walkington, N. Collars and intestines: Practical conforming Delaunay refinement. In *Proceedings of the 18th International Meshing Roundtable.* Springer, 2009, pp. 481–497.

[15] Rebay, S. Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal of Computational Physics 106*, 1 (1993), 125 – 138.

[16] Ruppert, J. A New and Simple Algorithm for Quality 2-dimensional Mesh Generation. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms* (Philadelphia, PA, USA, 1993), SODA '93, Society for Industrial and Applied Mathematics, pp. 83–92.

[17] Ruppert, J. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms 18*, 3 (1995), 548 – 585.

[18] Shewchuk, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds., vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[19] Shewchuk, J. R. *Delaunay Refinement Mesh Generation*. PhD thesis, Pittsburg, Pennsylvania, 1997.

[20] Üngör, A. Off-centers: A New Type of Steiner Points for Computing Size-optimal Guaranteed-quality Delaunay Triangulations. *Computational Geometry: Theory and Applications 42*, 2 (2009), 109–118.

Chapter 4

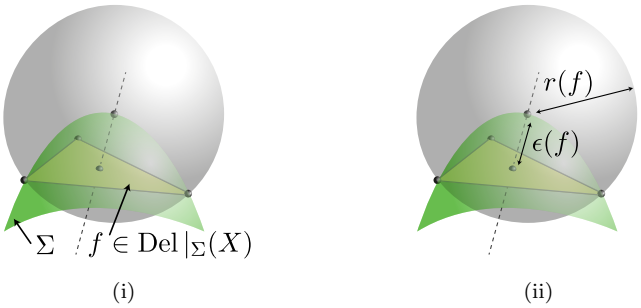
Surface Mesh Generation

In this chapter, I present a Frontal-Delaunay surface meshing algorithm for closed 2-manifolds embedded in $\mathbb{R}^3$. This new algorithm is an extension of the Frontal-Delaunay method developed for planar domains in Chapter 3 and is designed to combine the high-quality results achieved using advancing-front techniques with the provable bounds and theoretical guarantees of Delaunay-refinement schemes. The surface meshing algorithms presented in this chapter are based on the so-called *restricted* Delaunay triangulation – a subset of the full-dimensional Delaunay tessellation that conforms to the surface of interest. I review both the mechanics and theoretical development of existing restricted Delaunay-refinement algorithms before introducing the new Frontal-Delaunay strategy. Exploiting ideas similar to those presented in Chapter 3, I show that the use of 'off-centre' Steiner vertices, positioned along facets in the associated Voronoi complex, typically leads to an improvement in both the shape- and size-quality of the resulting surface tessellation. In addition to conventional shape-driven refinement methods, based on element radius-edge ratios, I show that a new size-optimal refinement strategy can be realised by positioning off-centre vertices such that a local mesh size function is satisfied. The use of this sizing function to generate graded meshes adhering to user defined size constraints is also explored. I use a simple theoretical model to prove termination and convergence for the proposed algorithm. I investigate the performance of the new Frontal-Delaunay strategy experimentally, and undertake a series of comparative studies, contrasting the performance of the new algorithm with a typical Delaunay-refinement technique. I demonstrate that the new Frontal-Delaunay method inherits many of the benefits of both Delaunay-refinement and advancing-front type methods, typically leading to the construction of very high quality triangulations in practice. Experiments are conducted using a range of complex benchmarks, verifying the robustness and practical performance of the proposed scheme. Work presented in this chapter appears in [11, 12].

4.1 Restricted Delaunay Refinement

Delaunay-refinement algorithms for surface meshing operate by incrementally introducing new Steiner vertices into an initially coarse Delaunay tessellation that 'conforms' to

Figure 4.1: Surface Delaunay ball for a restricted 2-face $f \in \text{Del}\,|_\Sigma(X)$, showing (i) the placement of the surface ball at the intersection of the dual Voronoi edge and the underlying surface Σ, and (ii) the radius $r(f)$ and surface discretisation error $\epsilon(f)$ associated with the surface ball.



the surface of interest. Contrary to the planar refinement techniques presented in Chapter 3, refinement schemes for surface meshing are designed not only to ensure that the resulting mesh satisfies element shape and size constraints, but that the geometry and topology of the mesh itself is an accurate piecewise approximation of the underlying surface. The surface meshing algorithms presented in this chapter are based on the so-called *restricted* Delaunay surface tessellation $\text{Del}\,|_\Sigma(X)$ – a sub-complex of the full-dimensional Delaunay tessellation $\text{Del}(X)$, containing the 2-faces $f \in \text{Del}(X)$ that *best* approximate an underlying surface Σ. The restricted Delaunay surface tessellation is introduced and defined in Section 2.2.

Before moving on to a detailed description of the meshing algorithms presented in this chapter, it is first necessary to redefine and introduce a number of important geometrical constructs that will be used throughout the developments that follow. Firstly:

Definition 4.1 (surface Delaunay ball)**.** Let $\text{Del}\,|_\Sigma(X)$ be a restricted Delaunay triangulation of a 2-manifold Σ. Given a 2-simplex $f \in \text{Del}\,|_\Sigma(X)$, any circumball of f centred on the surface Σ is a *surface Delaunay ball* of f, denoted $\text{SDB}(f)$.

Surface Delaunay balls are centred at the intersection of the associated Voronoi complex $\text{Vor}(X)$ with the surface Σ. Specifically, each 2-face $f \in \text{Del}\,|_\Sigma(X)$ is associated with an orthogonal edge in the Voronoi complex $v \in \text{Vor}(X)$, which is guaranteed, based on the properties of the restricted tessellation $\text{Del}\,|_\Sigma(X)$, to intersect with the surface Σ *at least* once. In some cases, especially when the sampling $X \in \Sigma$ is relatively coarse, there may be multiple surface balls associated with any given surface facet $f \in \text{Del}\,|_\Sigma(X)$. In such cases, it is typical to consider only the *largest* surface ball. See Figure 4.1 for an illustration of the surface Delaunay ball for a general facet f.

Definition 4.2 (surface discretisation error)**.** Let $\text{Del}\,|_\Sigma(X)$ be a restricted Delaunay triangulation of a 2-manifold Σ. Given a 2-simplex $f \in \text{Del}\,|_\Sigma(X)$, the *surface discretisation error* $\epsilon(f)$ is the Euclidean distance between the centres of the largest surface Delaunay ball of f and its diametric ball.

The surface discretisation error is a measure of how well the restricted triangulation $\text{Del}\,|_\Sigma(X)$ approximates the underlying surface Σ. Considering a surface facet

$f \in \text{Del}|_\Sigma(X)$, the surface discretisation error $\epsilon(f)$ is a measure of the distance between the face f and the furthest associated point on the surface Σ. This measure can be thought of as a one-sided discrete Hausdorff distance, $\epsilon(f) = \text{H}(\text{Del}|_\Sigma(X), \Sigma)$, defined from a set of representative points on the triangulation $\text{Del}|_\Sigma(X)$ to the surface Σ. Clearly, if the triangulation $\text{Del}|_\Sigma(X)$ is a good piecewise representation of the surface Σ, the surface discretisation error $\epsilon(f)$ should be small. See Figure 4.1 for a representation of the surface discretisation error for a facet $f \in \text{Del}|_\Sigma(X)$.

4.1.1 An Existing Algorithm

The development of provably-good Delaunay-refinement schemes for surface-based mesh generation is an ongoing area of research. In this section, I present an algorithm for the meshing of closed 2-manifolds embedded in $\mathbb{R}^3$, adapted largely from the methods presented by Boissonnat and Oudot in [4, 5]. This method is largely equivalent to the `CGALMESH` algorithm, available as part of the `CGAL` package, and summarised by Jamin, Alliez, Yvinec and Boissonnat in [15]. A similar algorithm is also outlined by Cheng, Dey and Shewchuk in [9]. I will refer to the algorithm presented in this section as the 'conventional' Delaunay-refinement approach, due to its direct use of circumcentre-based Steiner vertices.

In this study, I restrict my attention to so-called *remeshing* operations, in which the underlying surface Σ is specified as an existing manifold triangular complex $\mathcal{P}$. While this constraint may seem overly restrictive, it is important to recognise that the use of such triangulated surface representations is widespread within the computational modelling communities, associated with applications such as laser scanning, terrain modelling and some computer drawing applications. Furthermore, remeshing operations are a necessary task in many practical cases, with triangulated surface models often incorporating a number of undesirable defects, including elements of low shape quality and/or regions of unsuitable resolution. The remeshing algorithms presented in this study are designed to remedy these situations, generating new high-quality surface meshes that adhere to user defined sizing constraints. An extension of the methods presented here to more general classes of surfaces, such as parametric or implicit representations, is left for future investigations.

Following the approach described by Jamin et al. in [15], the Delaunay-refinement algorithm takes as input a surface domain, described by a closed 2-manifold Σ, an upper bound on the allowable element radius-edge ratio $\bar{\rho}$, a mesh size function $\bar{h}(\mathbf{x})$ defined at all points on the surface Σ and an upper bound on the allowable surface discretisation error $\bar{\epsilon}(\mathbf{x})$. The input surface Σ encloses a bounded volume Ω. The algorithm returns a triangulation $\mathcal{T}|_\Sigma$ of the surface Σ, where $\mathcal{T}|_\Sigma$ is a restricted Delaunay surface triangulation of a point-wise sampling $X \in \Sigma$, such that $\mathcal{T}|_\Sigma = \text{Del}|_\Sigma(X)$. As a by-product, the algorithm also returns a coarse triangulation $\mathcal{T}|_\Omega$ of the enclosed volume Ω, where $\mathcal{T}|_\Omega$ is a restricted Delaunay volume triangulation $\mathcal{T}|_\Omega = \text{Del}|_\Omega(X)$. Both $\text{Del}|_\Sigma(X)$ and $\text{Del}|_\Omega(X)$ are sub-complexes of the full-dimensional Delaunay tessellation $\text{Del}(X)$. Note that $\text{Del}|_\Sigma(X)$ is a triangular complex, while $\text{Del}|_\Omega(X)$ and $\text{Del}(X)$ are tetrahedral

Algorithm 4.1.1 Restricted Delaunay Surface Refinement

function DELAUNAYSURFACE($\Sigma, \Omega, \bar{\rho}, \bar{\epsilon}(\mathbf{x}), \bar{h}(\mathbf{x}), \mathcal{T}|_\Sigma, \mathcal{T}|_\Omega$)
  Form an initial sampling $X \in \Sigma$ such that X is well-spaced on Σ.
  Form Delaunay tessellation $\mathrm{Del}(X)$.
  for all ($\tau \in \mathrm{Del}(X)$) **do** ▷ {assemble restricted triangulations}
    Call RESTRICTEDDT($\tau, \mathrm{Del}\,|_\Sigma(X), \mathrm{Del}\,|_\Omega(X)$)
  end for
  Enqueue all 2-simplexes $Q|_\Sigma \leftarrow f \in \mathrm{Del}\,|_\Sigma(X)$. A 2-simplex f is enqueued if BADSIMPLEX2(f) returns TRUE.
  while ($Q|_\Sigma \neq \emptyset$) **do** ▷ {main refinement loop}
    Call REFINESIMPLEX2($f \leftarrow Q|_\Sigma$)
    for all (updated $\tau \in \mathrm{Del}(X)$) **do**
      Call RESTRICTEDDT($\tau, \mathrm{Del}\,|_\Sigma(X), \mathrm{Del}\,|_\Omega(X)$)
    end for
    Update $Q|_\Sigma$ to reflect changes to $\mathrm{Del}\,|_\Sigma(X)$.
  end while
  return $\mathcal{T}|_\Sigma = \mathrm{Del}\,|_\Sigma(X)$ and $\mathcal{T}|_\Omega = \mathrm{Del}\,|_\Omega(X)$
end function

function REFINESIMPLEX(f) ▷ {surface refinement}
  Call SURFACEDELAUNAYBALL($f, \mathrm{B}(\mathbf{c}, r)_{\max}$).
  Form new Steiner vertex $\mathbf{c}$ about $\mathrm{B}(\mathbf{c}, r)_{\max}$.
  Insert Steiner vertex $X \leftarrow \mathbf{c}$ and update $\mathrm{Del}(X) \leftarrow X$.
end function

function SURFACEDELAUNAYBALL(f) ▷ {surface ball}
  Form Voronoi edge $\mathbf{v}_f$ orthogonal to 2-simplex f.
  Form the set of associated surface Delaunay balls $\mathrm{B}(\mathbf{c}, r)_i$ for the restricted 2-simplex $f \in \mathrm{Del}\,|_\Sigma(X)$. Balls are centred about the set of surface intersections $\mathbf{v}_f \cap \Sigma \neq \emptyset$.
  return $\mathrm{B}(\mathbf{c}, r)_{\max}$, where $r_{\max}$ is maximal.
end function

function BADSIMPLEX(f) ▷ {termination criteria}
  return $(\rho(f) > \bar{\rho})$ or $\big(\epsilon(f) > \bar{\epsilon}(\mathbf{x}_f)\big)$ or $\big(h(f) > \bar{h}(\mathbf{x}_f)\big)$
end function

complexes. The Delaunay-refinement algorithm is summarised in Algorithm 4.1.1.

The Delaunay-refinement algorithm guarantees, firstly, that all elements in the output surface triangulation $f \in \mathcal{T}|_\Sigma$ satisfy both the element shape constraints, such that $\rho(f) \leq \bar{\rho}$, and the surface discretisation threshold, such that $\epsilon(f) \leq \bar{\epsilon}(\mathbf{x}_f)$. Furthermore, for sufficiently small mesh size functions $\bar{h}(\mathbf{x})$, the output surface triangulation $\mathcal{T}|_\Sigma$ is guaranteed to be a *good* piecewise approximation of the underlying surface Σ. Based on the properties of the restricted Delaunay triangulation outlined in Section 2.2, under such conditions it is known that $\mathcal{T}|_\Sigma$ is homeomorphic to Σ, that the Hausdorff distance $\mathrm{H}(\Sigma, \mathcal{T}|_\Sigma)$ is small, and that $\mathcal{T}|_\Sigma$ is a good geometric approximation to the surface Σ – with the properties of the triangulation converging toward the true normals, curvature and area of the underlying surface Σ as $\bar{h}(\mathbf{x}) \to 0$.

The Delaunay-refinement algorithm begins by creating an initial point-wise sampling of the surface $X \in \Sigma$. Exploiting the discrete representation available for Σ, the initial sampling can be obtained as a *well-distributed* subset of the existing vertices $Y \in \mathcal{P}$, where $\mathcal{P}$ is the polyhedral representation of the surface Σ. In the next step, the initial triangulation objects are formed. The full-dimensional Delaunay tessellation, $\mathrm{Del}(X)$, is built using the standard incremental framework outlined in Chapter 2. The restricted

surface and volume triangulations, $\text{Del}\,|_{\Sigma}(X)$ and $\text{Del}\,|_{\Omega}(X)$, are derived from $\text{Del}(X)$ by explicitly testing for intersections between the associated Voronoi complex $\text{Vor}(X)$ and the surface Σ. This process is discussed in further detail in subsequent sections. The main loop of the algorithm proceeds to incrementally refine any 2-faces $f \in \text{Del}\,|_{\Sigma}(X)$ that do not satisfy either the radius-edge, element size or surface discretisation requirements. The refinement process is priority scheduled, with triangles $f \in \text{Del}\,|_{\Sigma}(X)$ ordered according to their radius-edge ratios $\rho(f)$, ensuring that the element with the *worst* ratio is refined at each iteration. Individual elements are refined based on their surface Delaunay balls, with a triangle $f \in \text{Del}(X)$ eliminated by inserting the centre of the largest associated surface ball $\text{B}(\mathbf{c}, r)_{\text{max}} = \text{SDB}(f)$ into the tessellation $\text{Del}(X)$. This process is a direct generalisation of the circumcentre-based insertion method introduced in Ruppert's algorithm for planar domains, discussed in Chapter 3. As a consequence of changes to the full-dimensional tessellation following the insertion of a new Steiner vertex, corresponding updates to the restricted triangulations $\text{Del}\,|_{\Sigma}(X)$ and $\text{Del}\,|_{\Omega}(X)$ are instigated, ensuring that all tessellation objects remain in-sync throughout the refinement process. The Delaunay-refinement algorithm terminates when all 2-faces $f \in \text{Del}\,|_{\Sigma}(X)$ satisfy all radius-edge, size and surface discretisation thresholds, such that[1] $\rho(f) \leq \bar{\rho}$, $h(f) \leq \alpha \bar{h}(\mathbf{x}_f)$ and $\epsilon(f) \leq \bar{\epsilon}(\mathbf{x}_f)$, respectively, where the element size $h(f)$ is proportional to the radius of the associated surface ball $\text{B}(\mathbf{c}, r)_{\text{max}} = \text{SDB}(f)$, such that[2] $h(f) = \sqrt{3}\, r$, and $\bar{h}(\mathbf{x}_f)$, $\bar{\epsilon}(\mathbf{x}_f)$ are sampled at the centre of $\text{SDB}(f)$.

4.1.2 Discussion

Unlike the planar Delaunay-refinement algorithms presented in Chapter 3, the mechanics of the restricted surface Delaunay-refinement algorithm cannot be understood in terms of the effect of vertex insertion on element shape-quality alone. Instead, considerations of geometrical and topological fidelity impose additional constraints on the nature of the point-wise sampling itself, requiring that a set of geometric sizing constraints be satisfied. Before discussing the various geometrical and topological guarantees in detail, I first formalise the notion of *feature-size* for closed surface domains.

Definition 4.3 (medial-axis)**.** Given a bounded volumetric domain Ω enclosed by a surface Σ, the *medial-axis* of Ω is the topological closure of the set of centres of *empty* balls touching the surface Σ at more than one point.

Definition 4.4 (medial-distance)**.** Given a bounded volumetric domain Ω enclosed by a surface Σ, in addition to a point $\mathbf{x} \in \Sigma$, the *medial-distance*, denoted $d_M(\mathbf{x})$, is the Euclidean distance between $\mathbf{x}$ and the closest point on the medial-axis of Ω.

Noting that the medial-axis lies at the 'centre' of any geometrical features in the volumetric domain induced by the bounding surface Σ, it is clear that the associated

[1] The scalar $\alpha = \frac{4}{3}$, to ensure that the mean element size does not, on average, undershoot the target size $\bar{h}(\mathbf{x}_f)$, as per the discussions outlined in Section 3.2.

[2] The coefficient $\sqrt{3}$ represents the mapping between the edge length and diametric ball radius for an equilateral element. Such scaling ensures that size constraints are applied with respect to mean edge length.

medial-distance $d_M(\mathbf{x})$ is a measure of the local 'thickness' of the domain at any point $\mathbf{x} \in \Sigma$. Using such considerations, Boissonnat and Oudot [4, 5] introduce the notion of *loose* γ*-samples* to describe point-wise samplings $X \in \Sigma$ that induce restricted surface triangulations $\mathrm{Del}\,|_\Sigma(X)$ with favourable geometrical and topological guarantees.

Definition 4.5 (loose γ-sample[1])**.** Let Σ be a closed surface supporting a point-wise sampling $X \in \Sigma$. Let $\mathcal{E}_\mathrm{V}(X)$ be the set of edges in the associated Voronoi complex $\mathrm{Vor}(X)$. The sampling X is called a is called a *loose* γ*-sample* of Σ if: (i) for all points $\mathbf{x} \in \{\mathcal{E}_\mathrm{V}(X) \cap \Sigma\}$ the associated γ*-balls* are non-empty, such that $X \cap \mathrm{B}(\mathbf{x}, \gamma d_M(\mathbf{x})) \neq \emptyset$ for some $\gamma \in \mathbb{R}^+$, and (ii) the restricted triangulation $\mathrm{Del}\,|_\Sigma(X)$ includes at least one vertex on all connected components of the surface Σ.

'Loose' γ-samples differ from the standard γ-samples introduced by Amenta and Bern [1] according to the manner in which the underlying surface is 'interrogated'. Specifically, standard γ-samples require that for all points $\mathbf{x} \in \Sigma$ there exists *at least* one point $\mathbf{x}_i \in X$ within the local γ*-ball* centred at $\mathbf{x}$, such that $\|\mathbf{x} - \mathbf{x}_i\| \leq \gamma d_M(\mathbf{x})$. Such arguments require a *continuous* interrogation of the surface Σ, and are therefore problematic in the context of discrete meshing algorithms. The *loose* γ-samples of Boissonnat and Oudot seek to relax such constraints, by instead requiring that only a discrete set of γ-balls, centred at intersections between the edges of the Voronoi complex $\mathcal{E}_\mathrm{V}(X) \subseteq \mathrm{Vor}(X)$ and the surface Σ, are required to be non-empty. Recalling that the surface Delaunay balls associated with the 2-faces $f \in \mathrm{Del}\,|_\Sigma(X)$ are centred on such intersections, it is clear that the restricted Delaunay-refinement algorithm described in Section 4.1 can be used to construct loose γ-samples directly, provided that refinement continues until all surface balls $\mathrm{B}(\mathbf{c}, r)_{\max} = \mathrm{SDB}(f)$ satisfy local sizing constraints, such that $r \leq \gamma d_M(\mathbf{c})$.

Importantly, Boissonnat and Oudot have shown that restricted Delaunay triangulations built using loose γ-samples provide a 'good' approximation of the underlying surface, exhibiting a number of desirable geometrical and topological properties. I state these properties here without proof and instead refer the reader to the detailed expositions presented in [4, 5] for further details.

Proposition 4.1 (loose γ-sample, restricted triangulation)**.** *Let* Σ *be a closed surface, supporting a loose* γ*-sampling* $X \in \Sigma$*, with* $\gamma \leq 0.08$*. The associated restricted surface triangulation* $\mathrm{Del}\,|_\Sigma(X)$ *exhibits the following properties: (i) the triangulation* $\mathrm{Del}\,|_\Sigma(X)$ *is homeomorphic to the surface* Σ*, (ii) for any 2-face* $f \in \mathrm{Del}\,|_\Sigma(X)$ *the angle between the facet normal* $\hat{\mathbf{n}}_f$ *and the surface normal* $\hat{\mathbf{n}}_\Sigma$*, sampled at the vertices* $\mathbf{x} \in f$*, is* $O(\gamma)$*, and (iii) the one-sided Hausdorff distance* $\mathrm{H}(\mathrm{Del}\,|_\Sigma(X), \Sigma)$*, sampled at the centre of the surface Delaunay balls* $\mathrm{B}(\mathbf{c}, r)_{max} = \mathrm{SDB}(f)$ *associated with any 2-face* $f \in \mathrm{Del}\,|_\Sigma(X)$*, is* $O(\gamma^2)$*.*

In addition to considerations of geometrical and topological correctness, the preservation of adequate bounds on the shape quality of any surface facets $f \in \mathrm{Del}\,|_\Sigma(X)$ is also clearly a key requirement for the surface Delaunay-refinement algorithm described

[1] I adopt the term 'γ-sample' in preference to the original 'ϵ-sample' terminology of Boissonnat and Oudot to prevent confusion with the surface discretisation error, denoted $\epsilon(f)$ in this work.

in Section 4.1. Following an approach similar to that outlined in Chapter 3 for Ruppert's algorithm, bounds on the achievable element radius-edge ratios can be derived by ensuring that a minimum edge length is preserved throughout the refinement process.

Proposition 4.2 (refinement, shape-quality)**.** *Let the element-wise radius-edge threshold* $\bar{\rho} \geq 1$*. Given a low-quality surface facet* $f \in \mathrm{Del}\,|_\Sigma(X)$*, for which* $\rho(f) \geq \bar{\rho}$*, the minimum edge length* $\|\mathbf{e}_0\|$ *is not decreased following the insertion of a new Steiner vertex located at the centre of the largest surface Delaunay ball* $\mathrm{B}(\mathbf{c}, r)_{max} = \mathrm{SDB}(f)$ *associated with the facet* f*.*

Proof. Recalling the definition of the radius-edge ratio, shape-based refinement occurs when $\frac{r_d}{\|\mathbf{e}_0\|} \geq \bar{\rho}$, where r_d is the radius of the diametric disk associated with the surface facet f. Noting that the new Steiner vertex $\mathbf{c}$ lies on an edge in the associated Voronoi complex $\mathrm{Vor}(X)$, it is clear that there is no vertex $\mathbf{x}_i \in X$ lying closer to $\mathbf{c}$ than the vertices of the surface facet $\mathbf{x}_j \in f$, which are located on the circumference of the associated surface ball at a distance of r. As a consequence, the length of the of the shortest edge in the updated triangulation $\mathrm{Del}\,|_\Sigma(X \cup \{\mathbf{c}\})$ is bounded by the size of the surface ball, such that $\|\mathbf{e}_0{}'\| \geq r$, where $\|\mathbf{e}_0{}'\|$ is the minimum length of any new edges in the updated triangulation. Noting that the diametric ball of the facet f is never larger than the associated surface Delaunay ball[1], it is clear that $\|\mathbf{e}_0{}'\| \geq r_d$ and that $\|\mathbf{e}_0{}'\| \geq \bar{\rho}\,\|\mathbf{e}_0\|$ as a result. Given that $\bar{\rho} \geq 1$, it is clear that minimum edge length is not decreased by the refinement process. □

Recalling the relationship between the radius-edge ratio and element plane angles for triangular elements, such arguments show that an angle bound $\theta_{\min} = 30°$ can be achieved for the surface meshing case without compromising overall convergence. These bounds correspond to the idealised planar configuration, in which an infinite domain without boundaries is considered. Augmenting the surface refinement algorithm to handle embedded edge constraints, inscribed on the surface Σ, is left for future studies.

In addition to shape-based refinement, it is important to also consider the effect of both size- and surface-error-driven refinement strategies on the structure of the tessellation. Specifically, in such cases, a *high-quality* surface facet $f \in \mathrm{Del}\,|_\Sigma(X)$ with $\rho(f) \leq 1$ may be marked for further refinement if f is in violation of the local element size or surface error constraints. In such cases, while it is tolerable to accept *some* reduction in minimum edge length, it is important to ensure that such a reduction is bounded in the worst-case, guaranteeing that the resulting sampling X remains *well-distributed*, without pathologically small edges.

Proposition 4.3 (refinement, grading)**.** *Given a refinable surface facet* $f \in \mathrm{Del}\,|_\Sigma(X)$*, the minimum edge length* $\|\mathbf{e}_0\|$ *is decreased by a worst-case* $O(1)$ *factor following the insertion of a new Steiner vertex located at the centre of the largest surface Delaunay ball* $\mathrm{B}(\mathbf{c}, r)_{max} = \mathrm{SDB}(f)$ *associated with the facet* f*.*

[1] Given a surface facet $f \in \mathrm{Del}\,|_\Sigma(X)$ the radius of the associated surface Delaunay ball $\mathrm{B}(\mathbf{c}, r)_{\max} = \mathrm{SDB}(f)$ reaches a minimum when the underlying surface Σ is flat. In such cases, the surface ball and the diametric ball of f are equal. When the curvature of Σ is non-zero, the centre of the surface ball $\mathrm{SDB}(f)$ is offset from the plane of the facet f, and the radius is increased accordingly.

Proof. Following the same reasoning presented in Proposition 4.2, the insertion of a new Steiner vertex $\mathbf{c}$, positioned at the centre of the associated surface ball $\mathrm{B}(\mathbf{c}, r)_{\max} = \mathrm{SDB}(f)$, modifies the local distribution of edge lengths. Specifically, the minimum edge length $\|\mathbf{e}_0'\|$ in the updated triangulation $\mathrm{Del}\,|_\Sigma(X \cup \{\mathbf{c}\})$ can be expressed in terms of the radius of the associated surface ball, such that $\|\mathbf{e}_0'\| \geq r$. In the worst case, when the underlying surface Σ is flat, the new edge length can be related to the radius-edge ratio of f directly, satisfying $\|\mathbf{e}_0'\| \geq \rho(f)\|\mathbf{e}_0\|$. Noting that $\rho(f)$ achieves a minimum value when the facet f is equilateral, such that $\rho(f) = 1/\sqrt{3}$, it is clear that the minimum edge length is reduced by an $O(1)$ factor of $1/\sqrt{3}$ in the worst-case. □

Boissonnat and Oudot [4, 5] have shown that, when given a suitable mesh size function $h(\mathbf{x})$, the Delaunay-refinement algorithm described in Section 4.1 is guaranteed to produce high quality surface meshes that satisfy a variety of geometrical and topological constraints.

Proposition 4.4 (termination)**.** *Given a closed 2-manifold Σ, a radius-edge threshold $\bar{\rho} \geq 1$, positive mesh size and surface discretisation functions, $\bar{h}(\mathbf{x}) > 0$, $\bar{\epsilon}(\mathbf{x}) > 0$, defined for all $\mathbf{x} \in \Sigma$, the Delaunay-refinement algorithm (4.1) terminates in a finite number of steps.*

Proof. The Delaunay-refinement algorithm (4.1) refines any surface facet $f \in \mathrm{Del}\,|_\Sigma(X)$ if: (i) it is of poor shape quality, such that $\rho(f) \geq \bar{\rho}$, (ii) it is too large, such that $h(f) \geq \alpha\bar{h}(\mathbf{x}_f)$, or (iii) it violates the local surface discretisation error threshold, such that $\epsilon(f) \geq \bar{\epsilon}(\mathbf{x}_f)$.

- Case (i): Given a sufficiently large radius-edge threshold $\bar{\rho} \geq 1$ it is known, via Proposition 4.2, that refinement does not lead to a reduction in minimum edge length. Shape-based refinement therefore preserves the minimal edge length $\|\mathbf{e}_0\|_k$, associated with either the initial sampling X_0, or the insertion of some previous vertex $\mathbf{x}_k \in X$ due to size- or surface-error-driven refinement.
- Case (ii): Given that the mesh size function $\bar{h}(\mathbf{x})$ is positive, such that $\bar{h}(\mathbf{x}) \geq \bar{h}_0$ for some $\bar{h}_0 \in \mathbb{R}^+$, size-driven refinement is *declined* once the local element size is sufficiently small. Specifically, considering a limiting case, in which $h(f) = \sqrt{3}r = \alpha\bar{h}_0$, where r is the radius of the associated surface ball, Proposition 4.3 states that the minimum edge length is reduced by a factor of $1/\sqrt{3}$ in the worst-case. Rearranging the previous expressions, the minimum edge length is shown to be bounded above $(\alpha/3)\bar{h}_0$.
- Case (iii): Given that the surface discretisation threshold is positive, such that $\bar{\epsilon}(\mathbf{x}) \geq \bar{\epsilon}_0$ for some $\epsilon_0 \in \mathbb{R}^+$, it is clear that error-driven refinement is *declined* once the local surface discretisation error is sufficiently small. Noting that the surface discretisation error $\epsilon(f)$ cannot exceed the radius r of the associated surface ball, it is clear that $r \geq \bar{\epsilon}_0$ is a worst-case condition for surface-error-driven refinement. Considering the limiting case, Proposition 4.3 states that the minimum edge length is reduced by a factor of $1/\sqrt{3}$ in the worst-case, showing that minimal edge length is bounded above $(1/\sqrt{3})\bar{\epsilon}_0$.

Given the individual bounds on minimum edge length, the overall Delaunay-refinement algorithm (4.1) is known to preserve some positive minimum vertex separation distance $d = \min\big(\|\mathbf{e}_0\|_k, (\alpha/3)\bar{h}_0, (1/\sqrt{3})\bar{\epsilon}_0\big)$, and, via the Packing Lemma (3.1), is guaranteed to produce a point-wise sampling X of finite size. It is therefore clear that the algorithm inserts a bounded number of Steiner vertices and is guaranteed to terminate in a finite number of steps as a consequence. □

A proof of finite termination leads directly to a number of useful auxiliary guarantees and bounds on both the nature and the quality of the output tessellation. Adopting the conventional terminology, surfaces meshes generated using the Delaunay-refinement algorithm presented in Section 4.1 can be considered to be 'provably good'.

Corollary 4.1 (convergence)**.** *Given a closed 2-manifold Σ, a radius-edge threshold $\bar{\rho} \geq 1$, positive mesh size and surface discretisation functions, $\bar{h}(\mathbf{x}) > 0$, $\bar{\epsilon}(\mathbf{x}) > 0$, defined for all $\mathbf{x} \in \Sigma$, the restricted surface tessellation $\mathcal{T}|_\Sigma = \mathrm{Del}\,|_\Sigma(X)$ generated by the Delaunay-refinement algorithm (4.1) satisfies all constraints. Specifically, for all surface facets f in the output $\mathcal{T}|_\Sigma$: (i) all radius-edge ratios are bounded, such that $\rho(f) < \bar{\rho}$, (ii) all element sizes are bounded, such that $h(f) < \alpha\bar{h}(\mathbf{x}_f)$, where $\mathbf{x}_f$ is the centre of the associated surface ball $\mathrm{B}(\mathbf{c}, r)_{max} = \mathrm{SDB}(f)$ and $\alpha \in \mathbb{R}^+$, and (iii) all surface discretisation errors are bounded, such that $\epsilon(f) < \bar{\epsilon}(\mathbf{x}_f)$.*

Proof. Algorithm (4.1) maintains a queue of 'bad' elements throughout the refinement process, containing any surface facets $f \in \mathrm{Del}\,|_\Sigma(()X)$ that are in violation of one or more local constraints. Specifically, a given 2-face $f \in \mathrm{Del}\,|_\Sigma(X)$ is enqueued if: (i) $\rho(f) \geq \bar{\rho}$, (ii) if $h(f) \geq \alpha\bar{h}(\mathbf{x}_f)$, or (iii) if $\epsilon(f) \geq \bar{\epsilon}(\mathbf{x}_f)$. Given that the algorithm is known to terminate, it is clear that the refinement queue must become empty after a finite number of steps and that all entities in the resulting triangulation $\mathrm{Del}\,|_\Sigma(X)$ satisfy the requisite radius-edge ratio, element size and surface error constraints as a consequence. □

Based on the properties of the loose γ-samples of Boissonnat and Oudot, it is clear that the resulting surface triangulations generated using the Delaunay-refinement algorithm are guaranteed to be a good approximation of the underlying surface definition if the mesh size function is sufficiently small.

Corollary 4.2 (convergence, loose γ-sample)**.** *Given a closed 2-mainfold Σ and a γ-conforming size function $(\alpha/\sqrt{3})\bar{h}(\mathbf{x}) \leq \gamma d_M(\mathbf{x})$, the point-wise sampling $X \in \Sigma$ generated by the Delaunay-refinement algorithm (4.1) is a loose γ-sample. When $\gamma \leq 0.08$ the associated triangulation $\mathrm{Del}\,|_\Sigma(X)$ is an accurate topological and geometrical approximation of the surface Σ.*

Proof. The Delaunay-refinement algorithm (4.1) refines any large surface facets, ensuring that $h(f) \leq \alpha\bar{h}(\mathbf{x}_f)$ for all facets $f \in \mathrm{Del}\,|_\Sigma(X)$. Rearranging the previous expression, and substituting for the given mesh size constraints leads to $h(f) = \sqrt{3}r \leq \alpha\big(\sqrt{3}/\alpha\big)\gamma d_{\mathrm{M}}(\mathbf{x})$, which simplifies to $r \leq 0.08\, d_{\mathrm{M}}(\mathbf{x})$ given that $\gamma \leq 0.08$. Recalling Proposition 4.1, it is known that $\mathrm{Del}\,|_\Sigma(X)$ is an asymptotically good geometrical and topological approximation to the underlying surface Σ under such conditions. A list of topological guarantees and geometrical bounds are enumerated in Proposition 4.1. □

4.2 Restricted Frontal-Delaunay Methods

Frontal-Delaunay algorithms are a hybridisation of advancing-front and Delaunay-refinement techniques, in which a Delaunay triangulation is used to define the topology of the mesh while new Steiner vertices are inserted in a manner consistent with advancing-front methodologies. In practice, such techniques have been observed to produce very high-quality meshes, inheriting the smooth, semi-structured vertex placement of pure advancing-front methods and the optimal mesh topology of Delaunay-based approaches. While Frontal-Delaunay methods have previously been used by a range of authors in the context of planar, volumetric and parametric surface meshing, including, for example studies by Üngör and Erten [13], Rebay [19], Mavriplis [16], and Frey, Borouchaki, George [14], and Remacle, Henrotte, Carrier-Baudouin, Béchet, Marchandise, Geuzaine and Mouton [20], I am not aware of any previous investigations describing the application of such techniques to the surface meshing problem directly. The conventional advancing-front method, on the other hand, has been generalised to support direct surface meshing, as, for example, outlined in studies by Rypl [21, 22] and Schreiner, Scheidegger, Fleishman and Silva [23, 24].

In these previous studies, the conventional planar advancing-front methodology is directly extended to support surface operations. Meshing proceeds with the incremental introduction of a well distributed set of vertices X positioned on the surface $X \subseteq \Sigma$. Note that, contrary to the restricted Delaunay-refinement techniques introduced in previous sections, advancing-front methods typically maintain a partial manifold triangular complex $\mathcal{T}|_\Sigma$ only – they do not construct a full-dimensional tessellation $\mathcal{T}|_\Omega$. It should also be noted that, in addition to the usual limitations associated with advancing-front strategies, serious issues of robustness often afflict these techniques in the context of surface mesh generation specifically, due to the difficulties associated with reliably evaluating the requisite geometric intersection and overlap predicates for sets of non-planar elements. Additionally, for highly curved and/or poorly separated surface definitions it is difficult to ensure the topological correctness of the output mesh $\mathcal{T}|_\Sigma$. Schreiner et al. [23, 24] introduce a number of heuristic techniques in an effort to overcome these difficulties.

4.2.1 Off-centres

The new Frontal-Delaunay algorithm presented in this study is based on a generalisation of the 'off-centre' techniques previously developed for planar mesh generation, as discussed in detail in Chapter 3. In the planar meshing algorithm, new Steiner vertices are positioned along edges in the associated Voronoi diagram $\mathrm{Vor}(X)$. Consistent with the methods developed previously for the planar case, the placement of surface-based off-centre vertices is designed to satisfy both element size and shape constraints.

In this study, I consider the use of off-centre Steiner vertices to simulate the vertex placement strategy of a conventional advancing-front approach, while also preserving the framework of a Delaunay-refinement meshing algorithm. The aim of such a strategy is to recover the high element qualities and smooth, semi-structured meshes generated by frontal methods in practice, while inheriting the guaranteed bounds of Delaunay-

refinement based methods. Advancing-front algorithms typically incorporate a mesh size function $\bar{h}(\mathbf{x})$, a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^+$ defined over the domain to be meshed, where $\bar{h}(\mathbf{x})$ represents the desired edge length $\|e\|$ at any point $\mathbf{x} \in \Sigma$. This mesh size function typically incorporates size constraints dictated by the both the user and the geometry of the domain to be meshed. The construction of appropriate mesh size functions will be discussed in further detail in Section 4.4, but for now, it is sufficient to note that $\bar{h}(\mathbf{x})$ is a g-Lipschitz continuous function defined at all points on the surface to be meshed.

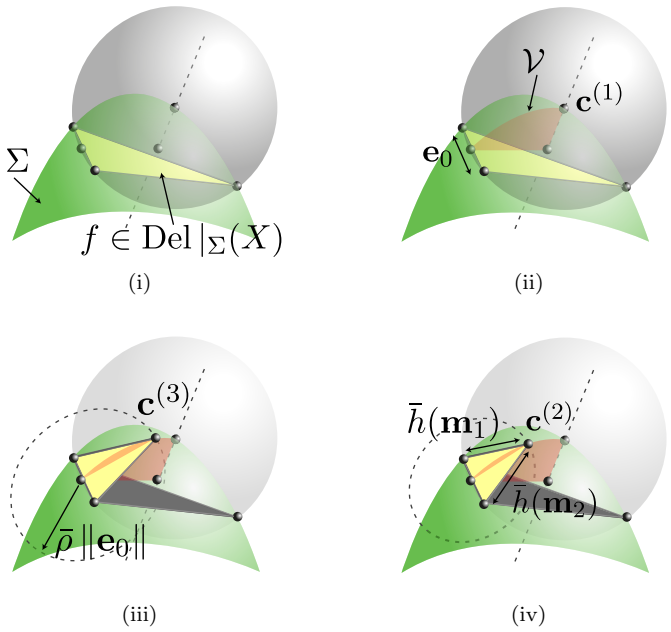
The proposed Frontal-Delaunay algorithm developed in this study is an extension of the restricted Delaunay-refinement algorithm presented in Section 4.1, modified to use off-centre rather than circumcentre-based refinement schemes. The basic framework of the algorithm is consistent with the Delaunay-refinement algorithm described previously, in which an initially coarse triangulation of a surface Σ is refined through the introduction of additional Steiner vertices $X \subseteq \Sigma$ until all element shape and size and surface-error constraints are satisfied. A conforming surface triangular complex $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$ is constructed through consideration of the set of restricted 2-faces of the full-dimensional tessellation $\text{Del}(X)$. A coarse conforming volumetric tetrahedral complex $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$ of the enclosed volume Ω is also constructed as a by-product. The constraints satisfied by the Frontal-Delaunay algorithm are identical to those described previously for the restricted Delaunay-refinement scheme, with upper bounds on the radius-edge ratio $\bar{\rho}$, surface discretisation error $\bar{\epsilon}(\mathbf{x}_f)$ and element size $\bar{h}(\mathbf{x}_f)$ all required to be satisfied for convergence. I refer the reader to Algorithm 4.1.1 for a detailed summary of the method.

4.2.2 Point-placement Strategy

Given a surface facet $f \in \text{Del}\,|_\Sigma(X)$ marked for refinement, the new Steiner vertex introduced to eliminate f is an off-centre, constructed based on local size and shape constraints. Adopting the *generalised* off-centre framework introduced by Üngör [25], the 'ideal' location of the off-centre $\mathbf{c}$ for the given element f is based on a consideration of the isosceles triangle σ formed about the short edge $\mathbf{e}_0 \in f$. I consider the locally-optimal placement of three points, $\mathbf{c}^{(1)}$, $\mathbf{c}^{(2)}$ and $\mathbf{c}^{(3)}$, designed to ensure that σ satisfies both local shape and size constraints. Type I vertices, $\mathbf{c}^{(1)}$, are located at the centre of element surface balls, and are used to satisfy constraints on the element radius-edge ratios. Type II vertices, $\mathbf{c}^{(2)}$, are *size-optimal* points, designed to satisfy element sizing constraints in a locally optimal fashion. Type III points, $\mathbf{c}^{(3)}$, are a generalisation of Üngör's off-centre vertices, and are designed to reduce the number of Steiner vertices required to satisfy the desired element radius-edge ratio. The final off-centre $\mathbf{c}$ is chosen as the point $\mathbf{c}^{(i)}$ that satisfies local constraints in a worst-case fashion. Point-placement diagrams for the generalised off-centre schemes are illustrated in Figure 3.1.

Given a *refinable* 2-simplex $f \in \text{Del}\,|_\Sigma(X)$, the size-optimal Type II vertex $\mathbf{c}^{(2)}$ is positioned at an intersection of the surface Σ and a plane $\mathcal{V}$, where $\mathcal{V}$ is aligned with the local face of the Voronoi diagram $\text{Vor}(X)$ associated with the frontal edge $\mathbf{e}_0 \in f$. The plane is positioned such that it passes through three local points on $\text{Vor}(X)$: the

Figure 4.2: Geometric constructions for the placement of off-centre Steiner vertices, showing (i) the surface Delaunay ball associated with a refinable 2-simplex $f \in \mathrm{Del}\,|_\Sigma(X)$, (ii) the plane $\mathcal{V}$, aligned with a local facet of the Voronoi diagram associated with the short edge $\mathbf{e}_0 \in f$, (iii) placement of the shape-optimal vertex $\mathbf{c}^{(3)}$ such that the required radius-edge ratio $\bar{\rho}$ is satisfied, and (iv) placement of the size-optimal vertex $\mathbf{c}^{(2)}$ such that local size constraints $\bar{h}(\mathbf{x}_f)$ are enforced.



midpoint of the frontal edge $\mathbf{e}_0 \in f$, the centre of the diametric ball of f and the centre of the surface Delaunay ball $\mathrm{B}(\mathbf{c}, r)_{\max} = \mathrm{SDB}(f)$. The vertex $\mathbf{c}^{(2)}$ is positioned such that the size of the new triangle $h(\sigma)$ satisfies local constraints. Specifically, the altitude of the triangle is calculated to ensure that the two new edges of σ are not too long, such that $\|\mathbf{e}_1\| \simeq \bar{h}(\mathbf{m}_1)$ and $\|\mathbf{e}_2\| \simeq \bar{h}(\mathbf{m}_2)$, where the $\mathbf{m}_i$'s are the new edge midpoints. Each constraint is solved for an associated altitude

$$a_i^{(2)} = \left(\bar{h}(\mathbf{m}_i)^2 - \|\tfrac{1}{2}\mathbf{e}_0\|^2\right)^{\frac{1}{2}}. \tag{4.1}$$

Given the altitudes, the position of the point $\mathbf{c}^{(2)}$ is calculated by computing the intersection of the surface Σ with a circle of radius $\frac{1}{2}(a_1^{(2)} + a_2^{(2)})$, centred at the midpoint of the frontal edge $\mathbf{e}_0 \in f$ and inscribed on the plane $\mathcal{V}$. In the case of multiple intersections, the candidate point $\mathbf{c}_i{}^{(2)}$ of closest alignment to the *frontal* direction vector $\mathbf{d}_f$ is selected. Specifically, the point $\mathbf{c}_i{}^{(2)}$ that maximises the scalar product $\left(\mathbf{c}_i{}^{(2)} - \mathbf{m}_0\right) \cdot \mathbf{d}_f$ is chosen, where $\mathbf{m}_0$ is the midpoint of the short edge $\mathbf{e}_0 \in f$ and the frontal direction vector $\mathbf{d}_f$ is taken from the midpoint $\mathbf{m}_0$ to the centre of the surface ball $\mathrm{B}(\mathbf{c}, r)_{\max} = \mathrm{SDB}(f)$.

For non-uniform $\bar{h}(\mathbf{x})$, expressions for the position of the point $\mathbf{c}^{(2)}$ are non-linear, with the altitudes a_i depending on an evaluation of the mesh size function at the edge midpoints $\bar{h}(\mathbf{m}_i)$ and visa-versa. In practice, since $\bar{h}(\mathbf{x})$ is guaranteed to be Lipschitz smooth, a simple iterative predictor-corrector procedure is sufficient to solve these expressions approximately. The positioning of size-optimal Type II Steiner vertices is illustrated

in Figure 4.2.

The shape-optimal Type III vertex $\mathbf{c}^{(3)}$ is placed following a generalisation of the methods introduced by Üngör [25], wherein $\mathbf{c}^{(3)}$ is positioned at an intersection of the surface Σ and a plane $\mathcal{V}$, where $\mathcal{V}$ is again a plane aligned with the local face of the Voronoi diagram $\mathrm{Vor}(X)$ associated with the frontal edge $\mathbf{e}_0 \in f$, as described previously. The vertex $\mathbf{c}^{(3)}$ is positioned to ensure that the shape of the new element σ satisfies the shape constraints, $\rho(\sigma) \leq \bar{\rho}$. Setting $\rho(\sigma) = \bar{\rho}$, leads to a solution for the *altitude* $a^{(3)}$ of σ, where

$$a^{(3)} = \tfrac{1}{2}\|\mathbf{e}_0\|\big(\tan\big(\tfrac{1}{2}\theta_{\min}\big)\big)^{-1}. \tag{4.2}$$

The position of the shape-optimal point $\mathbf{c}^{(3)}$ is then found by computing the intersection of the surface Σ with a circle of radius $a^{(3)}$, centred at the midpoint of the short edge $\mathbf{e}_0 \in f$ and inscribed on the plane $\mathcal{V}$. In the case of multiple intersections, the candidate point $\mathbf{c}_i^{(3)}$ of closest alignment to the *frontal* direction vector $\mathbf{d}_f$ is selected, as discussed previously. The positioning of shape-optimal Type III Steiner vertices is illustrated in Figure 4.2.

Using the size-optimal Type II point $\mathbf{c}^{(2)}$, the shape-optimal Type III point $\mathbf{c}^{(3)}$ and the standard Type I point $\mathbf{c}^{(1)}$, the final position of the refinement point $\mathbf{c}$ for the surface facet $f \in \mathrm{Del}\,|_{\Sigma}(X)$ is calculated. The point $\mathbf{c}$ is selected to satisfy the *limiting* local constraints, setting

$$\mathbf{c} = \begin{cases} \mathbf{c}^{(2)}, & \text{if } \big(d^{(2)} \leq d^{(1)}\big),\ \big(d^{(2)} \leq d^{(3)}\big) \text{ and } \big(d^{(2)} \geq \tfrac{1}{2}\|\mathbf{e}_0\|\big), \\ \mathbf{c}^{(3)}, & \text{if } \big(d^{(3)} \leq d^{(1)}\big), \\ \mathbf{c}^{(1)}, & \text{otherwise} \end{cases} \tag{4.3}$$

where the $d^{(i)} = \|\mathbf{c}^{(i)} - \mathbf{m}_0\|$ are distances from the midpoint of the frontal edge $\mathbf{e}_0$ to the Type I, Type II and Type III vertices, respectively. The cascading selection criteria is designed to ensure that the refinement scheme smoothly degenerates to that of a conventional circumball-based Delaunay-refinement strategy in limiting cases, while using locally size- or shape-optimal points where possible. Specifically, the conditions $d^{(2,3)} \leq d^{(1)}$ guarantee that $\mathbf{c}$ lies no further from the frontal edge $\mathbf{e}_0$ than the centre of the circumball associated with the facet f. Additionally, the condition $d^{(2)} \geq \frac{1}{2}\|\mathbf{e}_0\|$ ensures that the diametric ball of the edge $\mathbf{e}_0$ remains empty. Such behaviour guarantees that the size-optimal scheme is only selected when $\mathbf{e}_0$ is sufficiently small with respect to the local mesh size function – ensuring that $\mathbf{e}_0$ is a good *frontal* edge candidate in the context of a conventional advancing-front scheme. In all other cases, a shape-based strategy, guaranteed to reduce element radius-edge ratios, is selected.

4.2.3 Surface Intersections

In preceding discussions, the placement of off-centre Steiner vertices is achieved by locating the points of intersection between circles of specified radii and the underlying surface definition Σ – an idea inspired by the techniques of Schreiner et al. [24] in the development of their advancing-front algorithm `AFRONT`. These methods were originally

designed in the context of an advancing-front framework, and ensure that elements of the 'correct' Euclidean altitude are generated, even in regions of high surface curvature. As noted by Schreiner et al., such an approach is significantly more robust than alternative strategies based on iterative projection to the surface, guaranteeing that vertices can be reliably and accurately positioned in the neighbourhood of difficult geometric features, such as regions in which elements straddle sharp creases or ridges in Σ. I refer the reader to the presentations of Schreiner et al. [23, 24] for additional discussions and examples.

4.2.4 Discussion

The mechanics of the Frontal-Delaunay surface meshing algorithm are similar to those of the Delaunay-refinement scheme presented in Section 4.1. Specifically, since the Frontal-Delaunay algorithm involves a modification to the underlying point-placement strategy only, it inherits much of the same theoretical framework as the preceding Delaunay-refinement scheme. Importantly, being based on a restricted Delaunay paradigm, it too benefits from the topological and geometrical guarantees associated with the loose γ-samples of Boissonnat and Oudot [4, 5], as enumerated in Proposition 4.1. Differences in the point-placement strategies necessitate that a number of the propositions presented in Section 4.1 be re-evaluated for the new algorithm.

Proposition 4.5 (refinement, shape-quality)**.** *Let the element-wise radius-edge threshold $\bar{\rho} \geq 1$. Given a low-quality surface facet $f \in \mathrm{Del}|_{\Sigma}(X)$, for which $\rho(f) > \bar{\rho}$, the minimum edge length $\|\mathbf{e}_0\|$ is not decreased following either: (i) the insertion of a new Type I Steiner vertex, located at the centre of the largest surface Delaunay ball $\mathrm{B}(\mathbf{c}, r)_{max} = \mathrm{SDB}(f)$ associated with the facet f, or (ii) the insertion of a new Type III Steiner vertex, positioned on a facet of the associated Voronoi complex adjacent to the short edge in the facet f.*

Proof. Considering the two shape-driven vertex insertion strategies separately:

- Case (i): The bounds associated with the insertion of Type I Steiner vertices are presented in detail in Proposition 4.2, and I do not reproduce them here in full as a result. Specifically, recalling that the length of the new edges $\|\mathbf{e}_0'\|$ introduced by the insertion of a Type I vertex can be expressed in terms of the existing local minimum, such that $\|\mathbf{e}_0'\| \geq \bar{\rho}\,\|\mathbf{e}_0\|$, it is clear that the minimum length scale is preserved provided $\bar{\rho} \geq 1$.
- Case (ii): Type III Steiner vertices are positioned on a plane $\mathcal{V}$ aligned with the local facet of the underlying Voronoi complex associated with the short edge $e_0 \in f$. Based on the properties of the Voronoi complex, the plane $\mathcal{V}$ intersects e_0 at its midpoint. The Steiner vertex $\mathbf{c}^{(3)}$ is positioned to ensure that an isosceles triangle σ, formed between the existing vertices $\mathbf{x}_{i,j} \in e_0$ and $\mathbf{c}^{(3)}$, satisfies the desired shape-quality. Recalling that the point $\mathbf{c}^{(3)}$ lies on an adjacent facet of the Voronoi diagram, it is clear that the existing vertices $\mathbf{x}_{i,j} \in e_0$ are its closest neighbours. Considering the geometry of the triangle σ, the length of the new edges can be expressed in terms of the base length, such that $\|\mathbf{e}_0'\| = \frac{1}{2}\big(\sin\big(\frac{1}{2}\theta_{\min}\big)\big)^{-1}\|\mathbf{e}_0\|$, where $\theta_{\min}$ is the enclosed angle at the apex

of σ. Clearly, for $\theta_{\min} \leq 60°$, the existing minimum edge length $\|\mathbf{e}_0\|$ is preserved. Such a limit corresponds to an equivalent bound on the radius-edge ratio, such that $\bar{\rho} \geq 1/\sqrt{3}$.

Overall, it is clear that the insertion of Type I vertices is a limiting case, and that, as a result, any point-placement scheme based on the insertion of both Type I and Type III Steiner vertices requires $\bar{\rho} \geq 1$ to ensure that minimum edge length is preserved. Such a bound is consistent with the threshold given in Proposition 4.2. □

Proposition 4.6 (refinement, grading)**.** *Given a refinable surface facet $f \in \mathrm{Del}\,|_\Sigma(X)$, the minimum edge length $\|\mathbf{e}_0\|$ is decreased by an $O(1)$ factor in the worst-case following: (i) the insertion of a new Type I Steiner vertex located at the centre of the largest surface Delaunay ball $\mathrm{B}(\mathbf{c}, r)_{max} = \mathrm{SDB}(f)$ associated with the facet f, or (ii) the insertion of a new Type II Steiner vertex, positioned on a facet of the associated Voronoi complex adjacent to the shortest edge in the facet f.*

Proof. Considering the two size-driven vertex insertion strategies separately:

- Case (i): The bounds associated with the insertion of Type I Steiner vertices are presented in detail in Proposition 4.3, and I do not reproduce them here in full as a result. Specifically, recalling that in the worst-case, given an equilateral facet f, Type I refinement results in a new minimum edge length equal to the radius of the associated surface ball, such that the edge length is bounded above $(1/\sqrt{3})\|\mathbf{e}_0\|$.
- Case (ii): Type II Steiner vertices are positioned on a plane $\mathcal{V}$ aligned with the local facet of the underlying Voronoi complex associated with the short edge $e_0 \in f$. Based on the properties of the Voronoi complex, the plane $\mathcal{V}$ intersects e_0 at its midpoint. The Steiner vertex $\mathbf{c}^{(2)}$ is positioned to ensure that an isosceles triangle σ, formed between the existing vertices $\mathbf{x}_{i,j} \in e_0$ and $\mathbf{c}^{(2)}$, satisfies the desired local size constraints. In the worst-case, when $\bar{h}(\mathbf{x}) \leq \|\mathbf{e}_0\|$, the new vertex $\mathbf{c}^{(2)}$ is positioned at the intersection of $\mathcal{V}$ and the diametric ball associated with the edge e_0. Such a strategy ensures that the new edge length is bounded above $(1/\sqrt{2})\|\mathbf{e}_0\|$.

Overall, it is clear that the insertion of Type I vertices is a limiting case, and that, as a result, any point-placement scheme based on the insertion of both Type I and Type II Steiner vertices reduces minimum edge length by a factor of $(1/\sqrt{3})$ in the worst-case, consistent with the bounds given in Proposition 4.3. □

Importantly, the bounds given in Propositions 4.5 and 4.6 are identical to those derived for the standard Delaunay-refinement algorithm presented in Section 4.1, showing that the behaviour of the new Frontal-Delaunay and Delaunay-refinement algorithms is consistent in the worst-case. Specifically, the bounds stated in Propositions 4.5 and 4.6 show that the new Frontal-Delaunay algorithm is guaranteed to: (i) terminate in a finite number of steps, (ii) converge to a result satisfying all element shape-, size- and surface error constraints, and (iii) generate an output tessellation that is geometrically and topologically consistent with the input surface Σ, provided that the mesh size function is sufficiently small. A proof of these statements is identical to that presented previously

for the Delaunay-refinement algorithm (4.1), via Proposition 4.4 and Corollaries 4.1 and 4.2, and is therefore not reproduced in full here.

4.3 Calculation of Restricted Tessellations

The methods used to construct and maintain the restricted surface and volume triangulations $\text{Del}|_{\Sigma}(X) \subseteq \text{Del}(X)$ and $\text{Del}|_{\Omega}(X) \subseteq \text{Del}(X)$ are core components of the surface-based Delaunay meshing algorithms presented in this chapter. In this study, these objects are obtained from the full dimensional triangulation $\text{Del}(X)$ in an incremental fashion – evaluating the associated geometric predicates with respect to the surface Σ and volume Ω on an as-needed basis. All 2-simplexes $f \in \text{Del}(X)$ are associated with an edge in the associated Voronoi diagram $\mathbf{v}_f \in \text{Vor}(X)$, and any facet for which the intersection $\mathbf{v}_f \cap \Sigma \neq \emptyset$ are elements of the restricted surface triangulation $\text{Del}|_{\Sigma}(X)$. Likewise, all 3-simplexes $\tau \in \text{Del}(X)$ associated with an internal circumcentre $\mathbf{c} \in \Omega$ are elements of the restricted volume triangulation $\text{Del}|_{\Omega}(X)$. Clearly, these conditions can be checked incrementally, cycling through each element $\tau \in \text{Del}(X)$ one-by-one and performing the requisite geometric calculations. The incremental algorithm used to construct and maintain the restricted surface and volumetric triangulations is summarised in Algorithm 4.3.1.

The algorithm takes as input a 3-simplex τ, the full-dimensional tessellation $\text{Del}(X)$ and an existing pair of restricted surface and volumetric triangulation objects $\text{Del}|_{\Sigma}(X)$ and $\text{Del}|_{\Omega}(X)$. Considering that it is necessary to *maintain* rather than simply construct the various triangulation objects, support for both the addition of new elements $\tau \in \text{Del}(X)$ and the removal of recently deleted elements $\tau \notin \text{Del}(X)$ is provided. In the first case, where τ is a new element, the restricted triangulations are updated by performing the requisite geometric tests: (i) the element τ is added to $\text{Del}|_{\Omega}(X)$ if its circumcentre is fully enclosed by the surface Σ, and (ii) each of the 2-faces $f \in \tau$ are added to $\text{Del}|_{\Sigma}(X)$ if their associated Voronoi edges intersect with the surface Σ explicitly. In this study, the triangulated surface Σ is stored in a supporting AABB-tree, as outlined in Chapter 2, allowing these operations to be implemented via efficient tree-based traversals. In the second case, where τ is a recently deleted element, the restricted triangulations are updated based on a set of topological tests: (i) τ is removed from $\text{Del}|_{\Omega}(X)$ if $\tau \in \text{Del}|_{\Omega}(X)$, and (ii) the faces $f \in \tau$ are removed from $\text{Del}|_{\Sigma}(X)$ for any $f \in \text{Del}|_{\Sigma}(X)$. Recalling that, in the case of vertex insertion, the full-dimensional tessellation $\text{Del}(X)$ is updated using the Bowyer-Watson technique described in Chapter 2, it is clear that the efficient incremental management of the restricted surface and volume triangulations can be accomplished directly. Given an update to the full-dimensional tessellation $\text{Del}(X)$, a set of deleted elements $\tau_{\text{old}} \notin \text{Del}(X)$ and a set of newly created elements $\tau_{\text{new}} \in \text{Del}(X)$ are returned as output from the Bowyer-Watson process. The restricted triangulations are then updated in a two-pass process, first scanning through elements in the set τ_{old} and deleting any associated $\tau \in \text{Del}|_{\Omega}(X)$ and $f \in \text{Del}|_{\Sigma}(X)$ before subsequently iterating over elements in the set τ_{new} and adding any new $\tau \in \text{Del}|_{\Omega}(X)$ and $f \in \text{Del}|_{\Sigma}(X)$. Considering that, in the average-case, the sets τ_{old} and τ_{new} are of size $O(1)$, and, recalling

Algorithm 4.3.1 Incremental Restricted Delaunay Update

```
1: function RestrictedDT(τ, Del|_Σ(X), Del|_Ω(X))
2:     if (τ ∈ Del(X)) then                          ▷ {τ inserted into Del(X)}
3:         Call InsertFacets(τ, Del|_Σ(X), Del|_Ω(X))
4:     end if
5:     if (τ ∉ Del(X)) then                          ▷ {τ deleted from Del(X)}
6:         Call DeleteFacets(τ, Del|_Σ(X), Del|_Ω(X))
7:     end if
8: end function
```

```
1: function InsertFacets(τ, Del|_Σ(X), Del|_Ω(X))
2:     Form circumball B(c, r) for 3-simplex τ.
3:     if (c ∈ Ω) then                                ▷ {test enclosure}
4:         Update Del|_Ω(X) ← τ.
5:     end if
6:     for all (2-simplexes f ∈ τ) do
7:         Form Voronoi edge v_f orthogonal to 2-simplex f.
8:         if (v_f ∩ Σ ≠ ∅) then                      ▷ {test intersections}
9:             Update Del|_Σ(X) ← f.
10:        end if
11:    end for
12: end function
```

```
1: function DeleteFacets(τ, Del|_Σ(X), Del|_Ω(X))
2:     Delete τ from Del|_Ω(X), if τ ∈ Del|_Ω(X).
3:     for all (2-simplexes f ∈ τ) do
4:         Delete f from Del|_Σ(X), if f ∈ Del|_Σ(X).
5:     end for
6: end function
```

that the associated geometric predicates are implemented via traversals of a supporting AABB-tree, the expected complexity for a single such update is $O(\log(|\mathcal{P}|))$, where $\mathcal{P}$ is a triangulation of the underlying surface definition Σ.

4.4 Mesh Size Functions

The construction of high-quality mesh size functions is an important aspect of both the restricted Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 4.1 and 4.2. Good mesh size functions $\bar{h}(\mathbf{x})$ incorporate sizing constraints imposed by both the user and the geometry of the domain to be meshed. Geometric and user-defined contributions can be considered via two separate size functions, where $\bar{h}_u(\mathbf{x})$ represents user-defined sizing information and $\bar{h}_g(\mathbf{x})$ encapsulates sizing constraints dictated by the geometry of the domain. In this study, I require that both $\bar{h}_u(\mathbf{x})$ and $\bar{h}_g(\mathbf{x})$ be piecewise linear functions defined on a supporting *tetrahedral* complex $\mathcal{S}$ and that the supporting complex *cover* the domain to be meshed. These restrictions are typical of methods used in existing mesh generation algorithms. Construction of appropriate user-defined functions $\bar{h}_u(\mathbf{x})$ is highly problem specific and I do not discuss such methods in detail here. Consistent with the strategies outlined in Chapter 3, a range of analytical and solution-dependent functions $\bar{h}_u(\mathbf{x})$ can be defined considered.

The satisfaction of appropriate geometric constraints is especially important in the case of surface mesh generation, where it is known that the geometric and topological correctness of the triangulation is dependent on the nature of the underlying mesh size

function. Such considerations are discussed in detail in Section 4.1, where it is shown that $\bar{h}(\mathbf{x})$ is required to be sufficiently small to ensure that the resulting tessellation satisfies the loose γ-sampling constraints of Boissonnat and Oudot [6, 7]. Bounds on the local magnitude of the size function can be expressed in terms of the *medial-axis* and *local-feature-size* of the underlying domain:

Definition 4.6 (local feature size)**.** The local feature size $\text{lfs}(\mathbf{x})$ for a bounded domain Ω enclosed by a closed surface Σ is a function $f(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}^+$, where $f(\mathbf{x})$ is proportional to the minimum distance from any point $\mathbf{x} \in \Sigma$ to the *medial-axis* of Ω.

Specifically, Boissonnat and Oudot have shown that when $\bar{h}_g(\mathbf{x}) \leq 0.08\, d_{\text{M}}(\mathbf{x})$, where $d_{\text{M}}(\mathbf{x})$ is the minimum distance from a point $\mathbf{x} \in \Sigma$ to the medial-axis, the resulting surface tessellation $\text{Del}\,|_{\Sigma}(X)$ is guaranteed to be a good geometrical and topological approximation of the underlying surface Σ, as stated in Proposition 4.1. Such bounds are often found to be overly pessimistic, with surface triangulations generated using larger size functions, such that $\bar{h}_g(\mathbf{x}) = \gamma d_M(\mathbf{x})$, where $\gamma \geq 0.08$, often found to be geometrically and topologically correct in practice [9, Chapter 11].

4.4.1 Initial Size Estimates

Consistent with the methods presented for planar domains in Chapter 3, the geometric size function $\bar{h}_g(\mathbf{x})$ is constructed based on an approximation of the local feature size of the domain. In practice, since direct evaluation of the local feature size $\text{lfs}(\mathbf{x})$ is expensive, I construct $\bar{h}_g(\mathbf{x})$ as a piecewise linear approximation, formed from a set of sparse samples of $\text{lfs}(\mathbf{x})$. In the present study, the surface Σ is specified as an existing triangulated surface $\mathcal{P}$, allowing $\text{lfs}(\mathbf{x})$ to be sampled directly in a *well-distributed* fashion. I construct a sparse supporting complex $\mathcal{S}_g$ for $\bar{h}_g(\mathbf{x})$ by forming a coarse restricted Delaunay tessellation[1] $\text{Del}\,|_{\Omega}(Y)$ of the points in the original triangulated surface $Y \in \mathcal{P}$. Importantly, note that since the surface Σ is embedded in $\mathbb{R}^3$, such a tessellation is a tetrahedral complex, covering the enclosed volume Ω.

A discrete approximation to $\text{lfs}(\mathbf{x})$ can be calculated directly via the approach of Amenta and Bern [1, 2, 3, 10] which makes use of the sparse Voronoi complex $\text{Vor}(Y)$ associated with vertices in the supporting tessellation $Y \subseteq \mathcal{S}_g$. Given a point-wise sampling of the surface of the domain $Y \subseteq \Sigma$, an estimate for the local feature size is computed for each $\mathbf{y}_i \in Y$. This estimate is based on the distance from the vertex $\mathbf{y}_i$ to an approximation of the medial axis of the domain Ω. Consistent with the approach presented in Chapter 3 for planar domains, an approximation of the medial axis can be derived from the Voronoi complex $\text{Vor}(Y)$. The situation is somewhat more complicated for domains in $\mathbb{R}^3$, with only a *subset* of the Voronoi vertices providing a good approximation to the medial axis. The remaining Voronoi vertices may be positioned arbitrarily close to the surface Σ. This issue is ameliorated using a *filtering* approach due to Amenta, Bern and Kamvysselis [2], originally presented in the context of algorithms for robust *surface*

[1] In this thesis, I assume that the initial surface sampling $Y \in \Sigma$ is *sufficiently-dense*, ensuring that $\text{Del}\,|_{\Omega}(Y)$ is an accurate geometric and topological approximation to the domain Ω. The development of techniques to handle sparsely sampled domains is deferred for future investigations.

Algorithm 4.4.1 Evaluate Feature Size in $\mathbb{R}^3$

function FEATURESIZE3D($Y, \mathrm{Del}\,|_\Omega(Y), h_g(Y)$)
  Form the restricted tetrahedral complex $\mathrm{Del}\,|_\Omega(Y)$.
  for all ($\mathbf{y}_i \in Y$) **do** ▷ {evaluate size function at input vertices}
    for all $\left(\tau \in \mathrm{Del}\,|_\Omega(Y) \,|\, \tau \bigcap \mathbf{y}_i \neq \emptyset\right)$ **do**
      Evaluate the circumball $\mathrm{B}(\mathbf{c}, r)$ associated with the 3-simplex τ.
      Push the Voronoi vertex $\mathbf{v}_\tau = \mathbf{c}$ onto the set of local Voronoi vertices, $V_i \leftarrow \mathbf{v}_\tau$.
    end for
    Find the positive *pole* – the Voronoi vertex $\mathbf{v}^+ \in V_i$ maximising the distance $d^+_{\max} = \|\mathbf{y}_i - \mathbf{v}^+\|$.
    Find the negative *pole* – the Voronoi vertex $\mathbf{v}^- \in V_i$ maximising the distance $d^-_{\max} = \|\mathbf{y}_i - \mathbf{v}^-\|$ lying in the opposite halfspace, such that $\left(\mathbf{y}_i - \mathbf{v}^+\right) \cdot \left(\mathbf{y}_i - \mathbf{v}^-\right) < 0$.
    Estimate the minimum distance to the medial axis, setting $h_g(\mathbf{y}_i) = \min\left(d^+_{\max}, d^-_{\max}\right)$.
  end for
end function

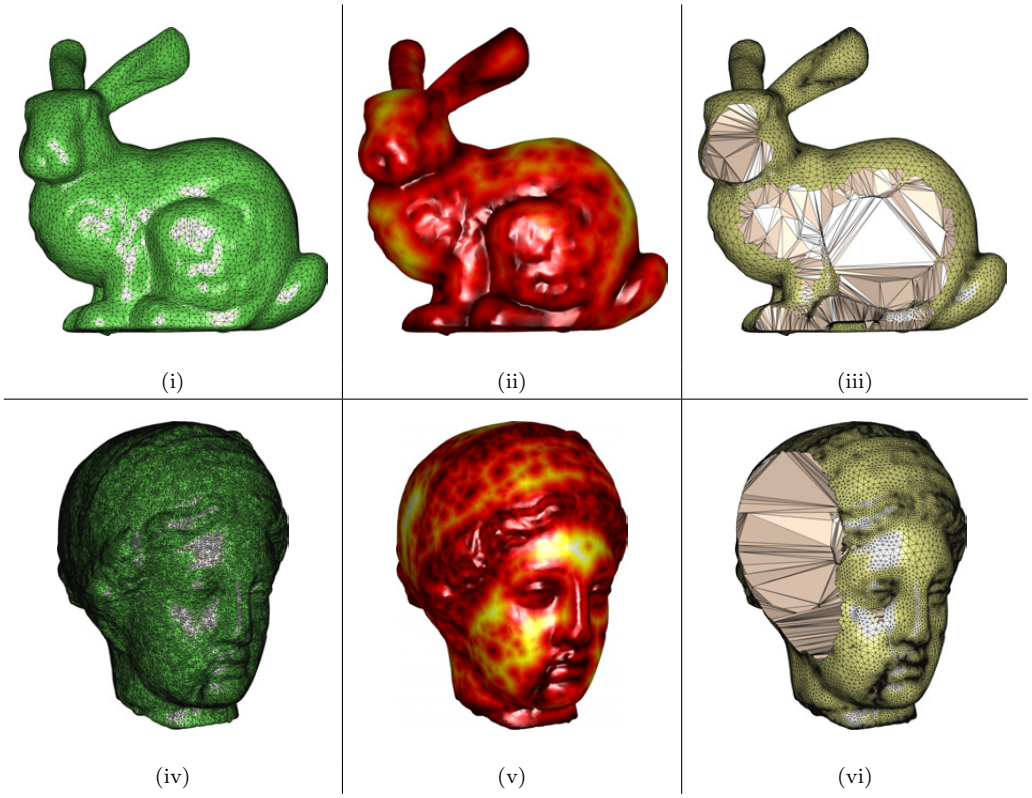
reconstruction. In such an approach, a pair of *poles* $\mathbf{v}_i^+, \mathbf{v}_i^-$ are identified for each sample $\mathbf{y}_i \in Y$, where $\mathbf{v}_i^+, \mathbf{v}_i^-$ are selected to be the furthest points in the Voronoi cell associated with the sample $\mathbf{y}_i$ and lying in either segment of the halfspace induced by Σ about the vertex $\mathbf{y}_i$. This *furthest-point* filtering ensures that Voronoi vertices lying close to the surface Σ are never used as an approximation to the medial axis. The local feature size at the vertex $\mathbf{y}_i$ is taken as the minimum distance to the poles $\mathbf{v}_i^+, \mathbf{v}_i^-$. This procedure is summarised in detail in Algorithm 4.4.1.

4.4.2 Sparse Refinement

By evaluating the local feature size at the set of surface samples $Y \in \Sigma$, only the boundary conditions for the full geometric size function $\bar{h}_g(\mathbf{x})$ are provided. A minimal refinement of the supporting tetrahedral complex $\mathcal{S}_g$ is undertaken, ensuring that the sizing information is smoothly extended into the interior of the bounded volume Ω. While such considerations are unnecessary for surface-only mesh generation, they are critical for the volumetric meshing operations presented in Chapter 5, and are detailed here for completeness.

Using a modified version of the restricted volumetric Delaunay-refinement algorithm, presented in detail in Section 5.1, a minimal refinement of the supporting tessellation $\mathcal{S}_g$ is performed, whereby any element $\tau \in \mathcal{S}_g$ is refined if it has a sufficiently large radius-edge ratio, such that $\rho(\tau) \geq \bar{\rho}_g$. Since only a minimal refinement is desired, a non-strict limit $\bar{\rho}_g$ is typically specified and in this study is set such that $\bar{\rho}_g = 4$. The resulting sparse complex $\mathcal{S}_g$ contains an approximation of the so-called *medial-axis* of the domain, in which a subset of the additional vertices added during refinement are equidistant to their associated closest features in $\mathcal{P}$. Importantly, it should be noted that since these points are positioned at the 'centre' of local geometrical features they coincide with ridges and peaks in the local feature size function $\mathrm{lfs}(\mathbf{x})$. The function values are initialised such that $\bar{h}_g(\mathbf{y}_j) = \infty$ for any new vertices $\mathbf{y}_j \in Y_{\text{new}}$, where Y_{new} is the set of Steiner vertices introduced during refinement. Appropriate mesh size values are calculated for these new

Figure 4.3: Geometric structures for the BUNNY and VENUS surface meshing problems. The triangulated surfaces Σ are shown (left), contours of element size function $\bar{h}(\mathbf{x})$ are shown (centre) and surface meshes $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$ obtained using the Frontal-Delaunay algorithm are shown (right). Cutaway views of surface meshes are shown, revealing the underlying restricted tetrahedral complexes $\text{Del}\,|_\Omega(X)$.



vertices via the subsequent *smoothing* operation, detailed in the following section.

A total mesh size function $\bar{h}(\mathbf{x})$ is ultimately obtained as a combination of the user-defined and geometric size functions, $\bar{h}_u(\mathbf{x})$ and $\bar{h}_g(\mathbf{x})$, such that $\bar{h}(\mathbf{x}) = \min\big(\bar{h}_u(\mathbf{x}), \bar{h}_g(\mathbf{x})\big)$. A supporting complex $\mathcal{S} = \text{Del}(Y_u \cup Y_g)$ is built for $\bar{h}(\mathbf{x})$, where the vertices $Y_u \in \mathcal{S}_u$ and $Y_g \in \mathcal{S}_g$ are supports for the user-defined and geometric size functions, respectively. It should be noted that the total size function $\bar{h}(\mathbf{x})$ expresses the limiting size constraints imposed by both user-defined and geometric inputs.

4.4.3 Size Function Smoothing

The final stage in the construction of the mesh size function $\bar{h}(\mathbf{x})$ involves the imposition of the Lipschitz constraint, g, via a so-called *gradient-limiting* process, such that, for any two points $\mathbf{x}_i, \mathbf{x}_j \in |\mathcal{P}|$, the local increase in size is bounded, with $\bar{h}(\mathbf{x}_j) \leq \bar{h}(\mathbf{x}_i) + g\,\|\mathbf{x}_i - \mathbf{x}_j\|$ and visa-versa. Such a function is said to be g-Lipschitz. Such limits on the smoothness of $\bar{h}(\mathbf{x})$ are introduced to ensure that the size constraints are consistent with desired element quality. Clearly, size functions that vary more slowly are expected to be congruent with improved element quality, while also leading to an increase in the number of elements in the output mesh $|\mathcal{T}|_\Sigma|$. Consistent with the approach presented in

Chapter 3 for planar domains, I adapt the efficient marching techniques introduced by Persson and Strang [17, 18] to impose smoothness constraints on the size function $h(\mathbf{x})$. This smoothing algorithm is summarised in detail in Algorithm 3.3.1.

4.4.4 Discussion

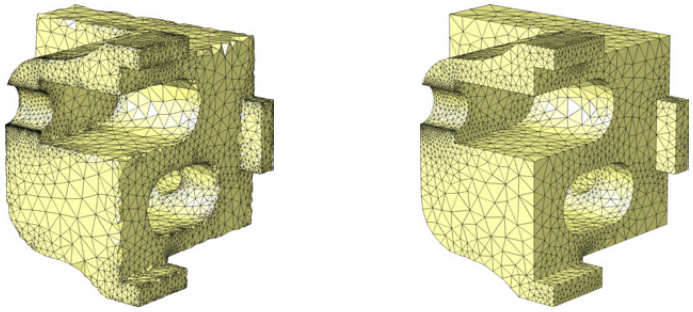
A set of geometric structures, including the triangulated surface definitions Σ, mesh size functions $\bar{h}(\mathbf{x})$ and resulting surface meshes $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$ are shown in Figure 4.3 for a pair of example problems. Note that the coarse volumetric tessellation $\text{Del}\,|_\Omega(X)$, generated as a by-product of the surface triangulation $\text{Del}\,|_\Sigma(X)$ is also shown in the cutaway views. In both cases, it can be seen that the mesh size functions $\bar{h}(\mathbf{x})$ reach local minima in: (i) regions of high curvature in the underlying surface Σ, and (ii) regions of small thickness in the underlying volume Ω. Additionally, the size function magnitude is seen to increase smoothly from these minima, reaching local maximum values in relatively flat regions that are well separated from areas of detail. The resolution of the corresponding surface meshes, in this case generated using the Frontal-Delaunay algorithm presented in Section 4.2, is clearly a good match to the contours of the associated mesh size functions, illustrating that the size-driven refinement schemes presented in previous sections is effective.

4.5 Domains with Sharp Features

Special care is needed to ensure that the Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 4.1 and 4.2 faithfully re-mesh domains containing sharp ridges and creases in the underlying surface Σ, and, furthermore, that the convergence of the meshing algorithms is preserved in such cases. While the presence of *isolated* sharp ridges in the surface Σ does not typically limit the achievable shape-quality of elements in the resulting surface meshes $\text{Del}\,|_\Sigma(X)$, complex configurations, in which multiple ridges and creases meet at small angles, can impose such element shape restrictions. Consistent with the discussions presented in Chapter 3 concerning the behaviour of the planar meshing algorithms, use of the unmodified Delaunay-refinement or Frontal-Delaunay algorithms in such cases can lead to non-convergence. Additionally, even in cases where convergence is obtained, the reproduction of sharp features in the resulting surface meshes $\text{Del}\,|_\Sigma(X)$ is typically poor, with Steiner vertices generally positioned away from the apex of any ridges or creases, leading to a noisy 'rounding-off' of sharp features in the underlying surface Σ.

In the present study I adopt a strategy inspired by the so-called *protecting balls* approach presented by Cheng, Dey and Ramos in [8]. Firstly, recalling that both the Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 4.1 and 4.2 start from an initial point-wise sampling X_s of the surface Σ, special care is taken to ensure that X_s includes an explicit sampling of any sharp features in the surface Σ. In practice, a single vertex is added to X_s for any sharp peaks in Σ, while a one-dimensional discretisation of any ridges or creases is performed, guaranteeing that edges aligned with

Figure 4.4: Illustration of surface mesh generation for domains containing sharp features, showing (i) results generated using the standard Frontal-Delaunay algorithm without treatment of sharp features, illustrating 'rounding' effects at ridges, and (ii) equivalent results using the protecting balls strategy.



(i) (ii)

these features satisfy local sizing constraints. While such a strategy ensures that sharp features are correctly incorporated within the initialisation phase, additional protection is necessary to ensure that such features correctly survive the refinement process. The protecting balls strategy encloses any sharp features in the domain within a set of Euclidean balls, within which refinement, through the introduction of new Steiner vertices, is prohibited. The Delaunay-refinement and Frontal-Delaunay methods presented in Sections 4.1 and 4.2 are modified to test the suitability of any candidate Steiner vertices, declining to perform refinements that would result in the introduction of new vertices within a protecting ball. Such filtering is accomplished efficiently by storing the collection of protecting balls in a supporting AABB-tree. Due to the rejection of Steiner vertices, the protecting balls strategy can have a negative effect on element shape-quality, allowing surface facets that would otherwise be refined to persist in the final output mesh $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$. Note though, that the protecting balls are active in only a narrow region adjacent to sharp features, and that all element constraints are fully satisfied in the remainder of the domain. The effectiveness of the protecting balls strategy is illustrated in Figure 4.4, where the same domain is meshed with and without the protection of sharp features. Clearly, the protecting balls strategy is successful not only in recovering the sharp ridges in the underlying surface, but also in preventing any spurious over-refinement, generated by the standard Frontal-Delaunay algorithm in an effort to satisfy the surface discretisation error thresholds in the vicinity of the sharp creases in the domain.

Importantly, the full protecting balls strategy presented by Cheng et al. in [8], also makes use of the so-called *weighted* Delaunay tessellation, a variation of the conventional Delaunay structure that incorporates non-negative scalar vertex weights that can be used to influence the topology of the resulting tessellation. Cheng et al. use such a structure to guarantee that adjacent vertices positioned along sharp ridges in the surface Σ are connected by an edge in the weighted, restricted triangulation $\text{Del}\,|_\Sigma(X)$, ensuring fidelity to the underlying domain. In the present work, I have utilised a conventional Delaunay tessellation only, admitting the potential for spurious topological connections

in the neighbourhood of sharp features. In practice, such situations have not been found to occur, based on the comprehensive set of benchmark meshes presented in Section 4.6. The use of a weighted tessellation, as described in [8, 9] and [15] is an important avenue for future development.

4.6 Results & Discussion

The performance of the surface meshing algorithms introduced in this chapter was investigated experimentally, with both the Delaunay-refinement and Frontal-Delaunay algorithms developed in Sections 4.1 and 4.2 used to mesh a series of fifteen benchmark problems of varying size and complexity. A range of test domains were chosen from a diverse set of application areas, including problems from computational modelling and simulation, computer graphics and medical imaging. Meshes for each domain, generated using the Frontal-Delaunay approach, are shown in Figure 4.6. Note that a number of test-cases include sharp surface ridges. Both the Frontal-Delaunay and Delaunay-refinement algorithms described in Sections 4.1 and 4.2 have been implemented, allowing the performance and output of the two algorithms to be compared side-by-side. Due to the similarities in structure, a common code-base was used, with the algorithms differing only in the type of Steiner vertices inserted and in the manner in which the queue of bad triangles is updated, as discussed in Section 4.2. Both algorithms are built using the `TRIPOD` and `LUMBERJACK` packages – making use of the efficient incremental Delaunay triangulation and spatial indexing frameworks presented in Chapter 2. Additionally, updates to the restricted surface and volumetric triangulations are achieved via an implementation of Algorithm 4.3.1, where the geometric predicates are evaluated using double precision arithmetic. The Frontal-Delaunay and Delaunay-refinement algorithms are included in `JIGSAW` – a new mesh generation library built using the algorithms developed in this thesis. Both algorithms are implemented in C++ and compiled as 64-bit executables.

Meshes generated using the new Frontal-Delaunay algorithm for the full set of benchmarks are presented in Figure 4.6, demonstrating the effectiveness of the new strategy in practice. These meshes are generated using relatively coarse element size constraints, with the mesh size functions $\bar{h}(\mathbf{x})$ constructed using a Lipschitz smoothness parameter $g = (3/10)$. Tight bounds on element shape quality are imposed, setting a radius-edge ratio $\bar{\rho}$ equivalent to $\theta_{\min} \simeq 29^\circ$ and $\theta_{\max} \simeq 122^\circ$. A non-uniform surface discretisation threshold was enforced, setting $\bar{\epsilon} = \beta\bar{h}(\mathbf{x})$, with $\beta = (1/10)$. For $\beta < 1$, such a setting ensures that the discrete Hausdorff associated with each surface facets $f \in \mathcal{T}|_\Sigma$ is a fraction of the local edge length. Qualitatively, the results shown in Figure 4.6 confirm that the new Frontal-Delaunay algorithm is successful in practice, generating high-quality meshes for a series of complex inputs. It can be seen that graded meshes are generated in all cases, with highly resolved regions in areas of high curvature smoothly transitioning to sparser representations throughout the remainder of the domains. It is also noted that convergence is achieved in all cases, confirming that the new point-placement schemes do not negatively impact on practical performance. Additionally, it is important to note that convergence is achieved for a number of inputs containing sharp ridges, confirming

Figure 4.5: Benchmark problems for surface meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = (3/10)$. A constant radius-edge ratio limit is specified, such that $\theta_{\min} \simeq 29^\circ$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element area-length ratio $a(f)$, plane angle $\theta(f)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

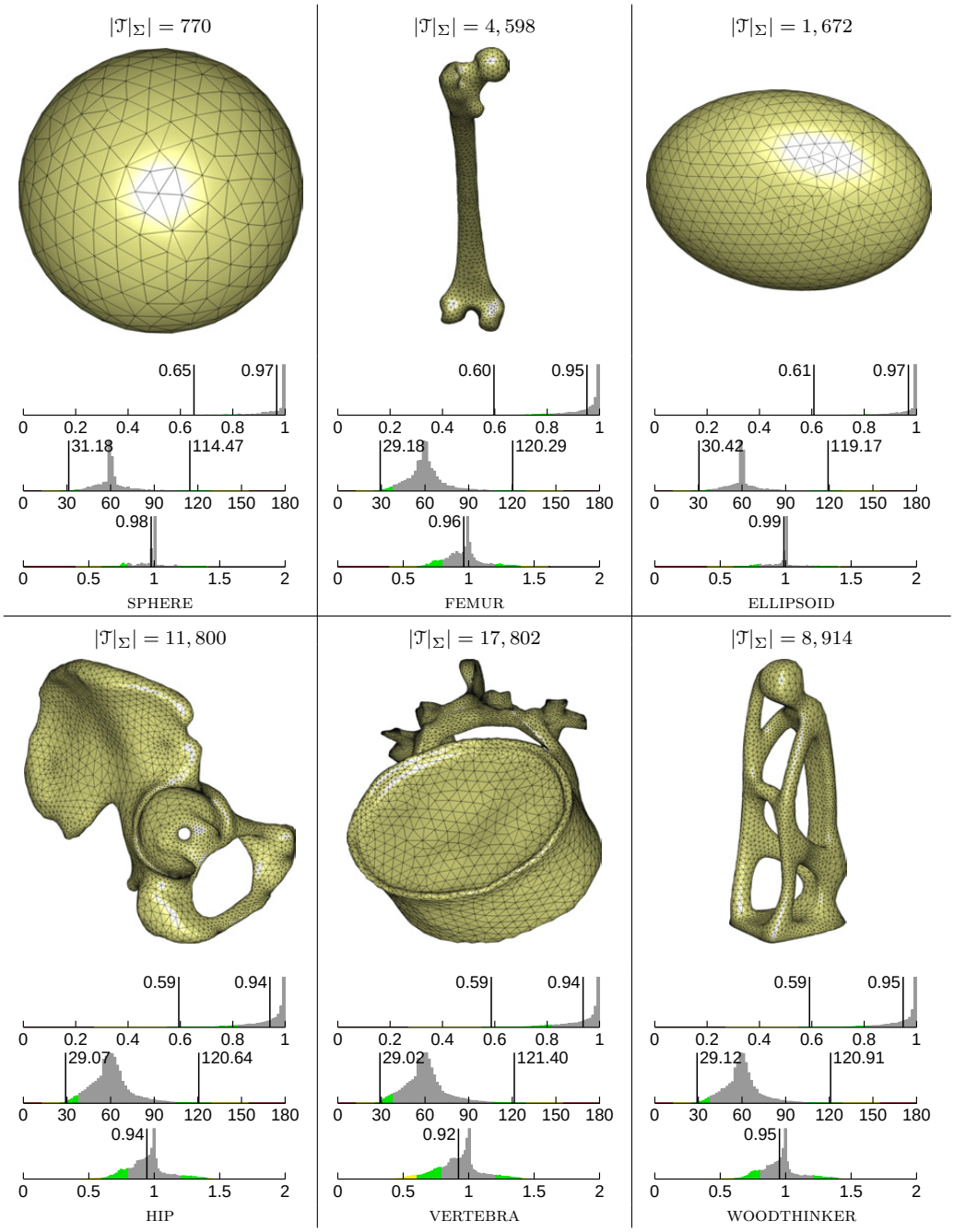


Figure 4.6: Benchmark problems for surface meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = (3/10)$. A constant radius-edge ratio limit is specified, such that $\theta_{\min} \simeq 29°$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element area-length ratio $a(f)$, plane angle $\theta(f)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

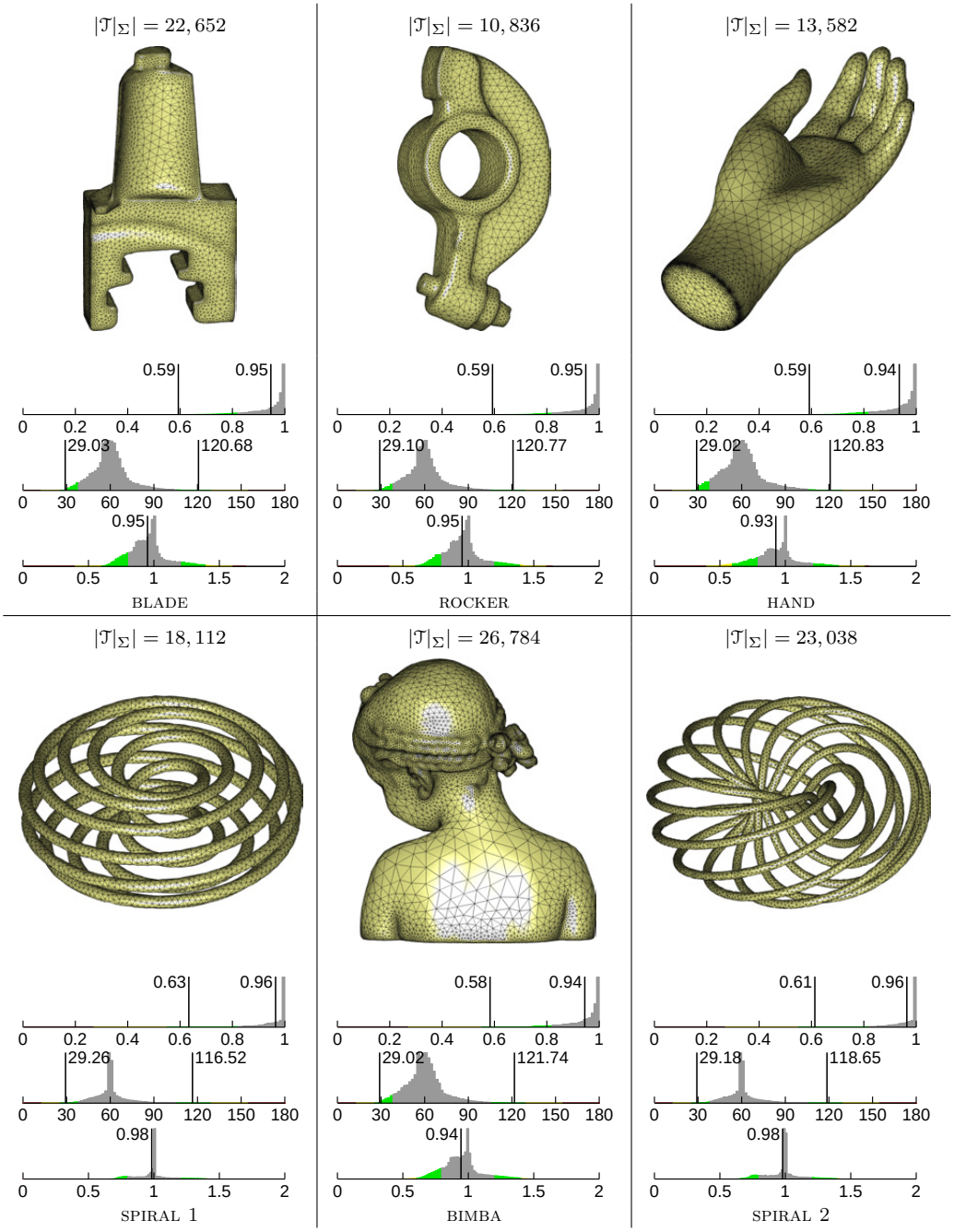
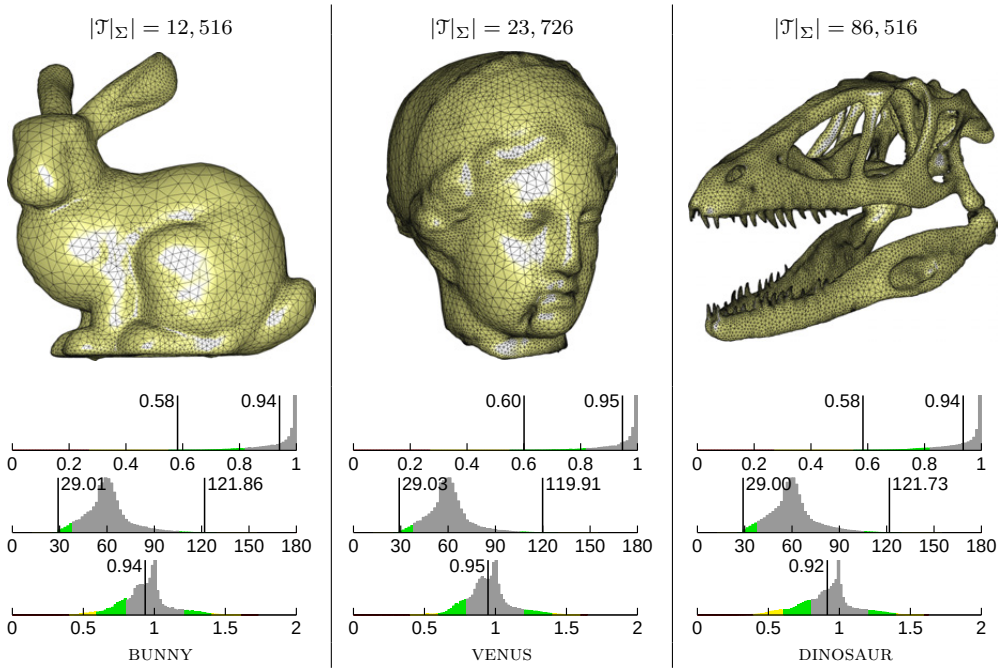


Figure 4.7: Benchmark problems for surface meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = (3/10)$. A constant radius-edge ratio limit is specified, such that $\theta_{\min} \simeq 29^\circ$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element area-length ratio $a(f)$, plane angle $\theta(f)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.



the effectiveness of the *protecting-balls* strategy for such domains.

4.6.1 Comparative Performance

The results of a comparative performance study, contrasting the effectiveness of the Frontal-Delaunay and Delaunay-refinement meshing algorithms, is presented in Figures 4.8 and 4.9 and includes detailed results for the BUNNY and VENUS test problems. Specifically, the effectiveness of the new size-driven refinement scheme is addressed, comparing a range of meshes generated using the Frontal-Delaunay and Delaunay-refinement algorithms for a range of different input parameters. The resulting meshes are compared in terms of their size- and shape-quality, and their underlying structure. In addition to the detailed results presented for the BUNNY and VENUS problems, a simplified set of comparisons are also presented for the full set of benchmark problems.

4.6.1.1 Size-driven Refinement

A detailed study of meshes generated for the BUNNY and VENUS problems, presented in Figures 4.8 and 4.9, examines the impact of the size-optimal Type II off-centres introduced in Section 4.2. A set of meshes are generated for test cases using low, medium and high resolution settings, where the associated mesh size functions $\bar{h}(\mathbf{x})$ are constructed using Lipschitz smoothness values of $g = (3/10)$, $g = (2/10)$ and $g = (1/10)$, respectively. The

radius-edge ratio limit is held constant across all test cases, such that $\theta_{\min} \simeq 29^\circ$. A non-uniform surface discretisation threshold is enforced, setting $\bar{\epsilon} = \beta \bar{h}(\mathbf{x})$, with $\beta = (1/10)$. For all test problems, element quality has been catalogued, with normalised histograms of element area-length $a(f)$, plane angle $\theta(f)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean area-length measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length. Note that the VENUS test case includes a number of sharp ridges in the underlying surface Σ, located near the neck of the statue.

Analysis of Figures 4.8 and 4.9 shows that both the Frontal-Delaunay and Delaunay-refinement algorithms generate high-quality meshes for all test cases – satisfying the required element angle thresholds. Inspection of distributions of $a(f)$, $\theta(f)$ and $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ show that the Frontal-Delaunay algorithm consistently outperforms the Delaunay-refinement scheme, generating meshes with higher mean area-length ratios and 'tight' distributions of element plane angle and relative edge length in all cases. The most significant difference in behaviour between the two algorithms appears to be in the way that mesh size constraints are imposed. Given the narrow distributions of relative edge length, strongly clustered about $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$, it is clear that the Frontal-Delaunay algorithm successfully generates meshes accurately conforming to the imposed mesh size constraints. In contrast, output generated using the Delaunay-refinement scheme is seen to incorporate significant sizing error, typified by 'broad' distributions of relative edge length straddling $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$. While the mean relative edge lengths are comparable in all cases, the Frontal-Delaunay algorithm clearly generates much more accurate output on an element-by-element basis.

Additionally, it is evident that the quality of meshes generated using the Frontal-Delaunay algorithm improves as $g \to 0$, as indicated by the increase in mean $a(f)$, the narrowing of the $\theta(f)$ distribution about 60.0°[1] and the narrowing of the distribution of relative edge length about $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$. In contrast, similar analysis shows that the Delaunay-refinement results are essentially independent of the magnitude of the mesh size function, $\bar{h}(\mathbf{x})$, with distributions of $a(f)$, $\theta(f)$ and $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ seen to be broadly consistent across all cases. Visually, the enhanced quality of the meshes generated using the Frontal-Delaunay algorithm is evident, with marked increases in both smoothness and sub-structure. The meshes generated by both algorithms are of similar sizes, with neither algorithm showing a consistent deviation in terms of element count $|\mathcal{T}|_\Sigma|$.

Analysis of the Steiner refinement strategies logged throughout this study show that the majority of the off-centres chosen by the Frontal-Delaunay algorithm are either size-optimal Type II points ($\simeq 60\%$) or conventional Type I circumcentres ($\simeq 40\%$), exhibiting similar ratios to those observed for the planar algorithm presented in Chapter 3. The bias toward size-optimal off-centres is also observed to increase as $h(\mathbf{x}) \to 0$. The use of shape-optimal Type III off-centres is rare ($\ll 1\%$). These results are not surprising, simply indicating that when the magnitude of the mesh size function $\bar{h}(\mathbf{x})$ is sufficiently

[1] Triangles with ideal shape quality are equilateral, explaining why $\theta(f) \to 60.0^\circ$ as shape-quality is improved.

Figure 4.8: Size-driven refinement study for the BUNNY problem, showing meshes generated using the Frontal-Delaunay (upper) and Delaunay-Refinement (lower) algorithms. Meshes are generated with increasing values of size function smoothness, $g = (3/10)$, $g = (2/10)$ and $g = (1/10)$, from left to right, respectively. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29^\circ$. A non-uniform surface discretisation threshold is imposed, such that $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element area-length ratio $a(f)$, plane angle $\theta(f)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

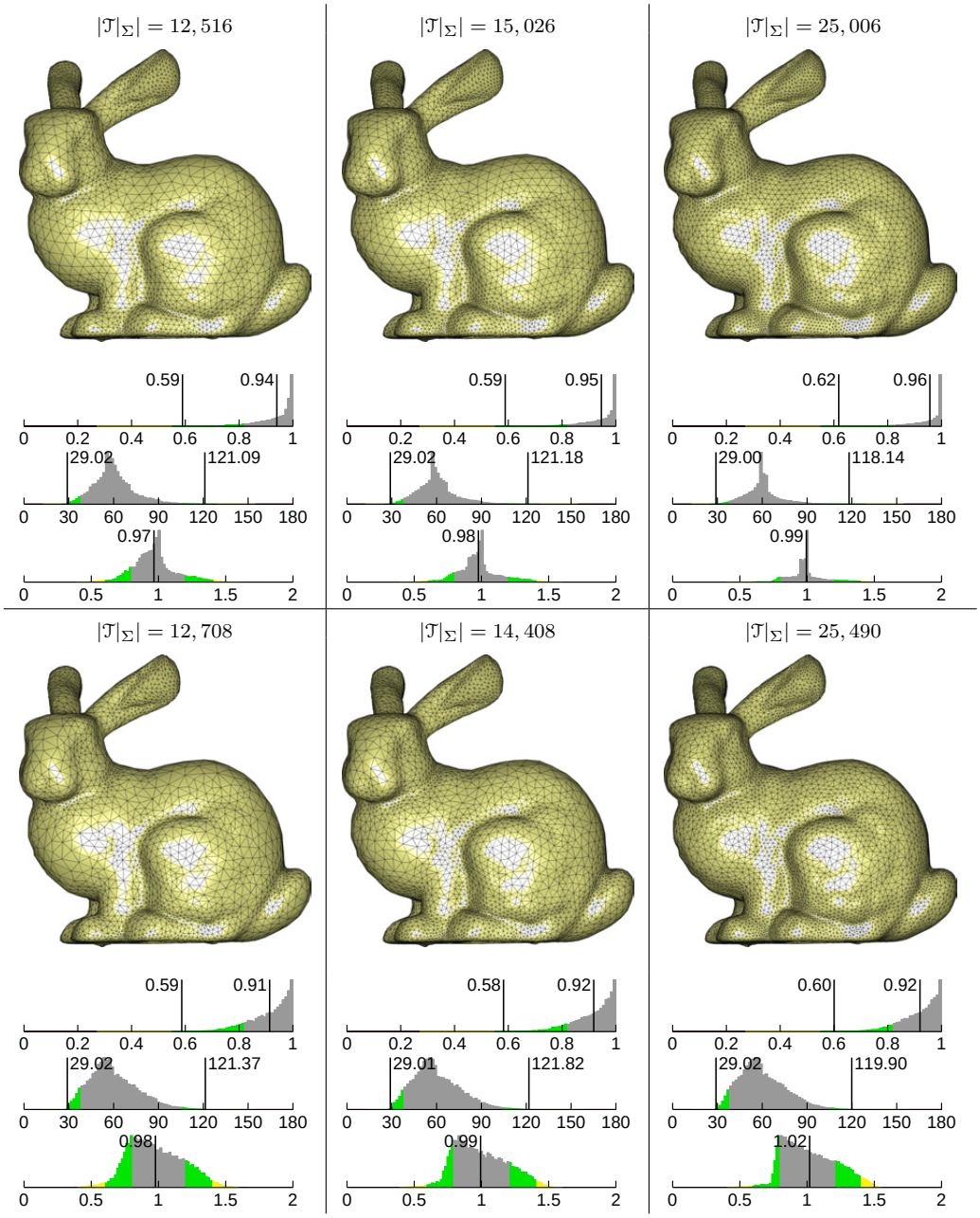


Figure 4.9: Size-driven refinement study for the VENUS problem, showing meshes generated using the Frontal-Delaunay (upper) and Delaunay-Refinement (lower) algorithms. Meshes are generated with increasing values of size function smoothness, $g = (3/10)$, $g = (2/10)$ and $g = (1/10)$, from left to right, respectively. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29^\circ$. A non-uniform surface discretisation threshold is imposed, such that $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element area-length ratio $a(f)$, plane angle $\theta(f)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

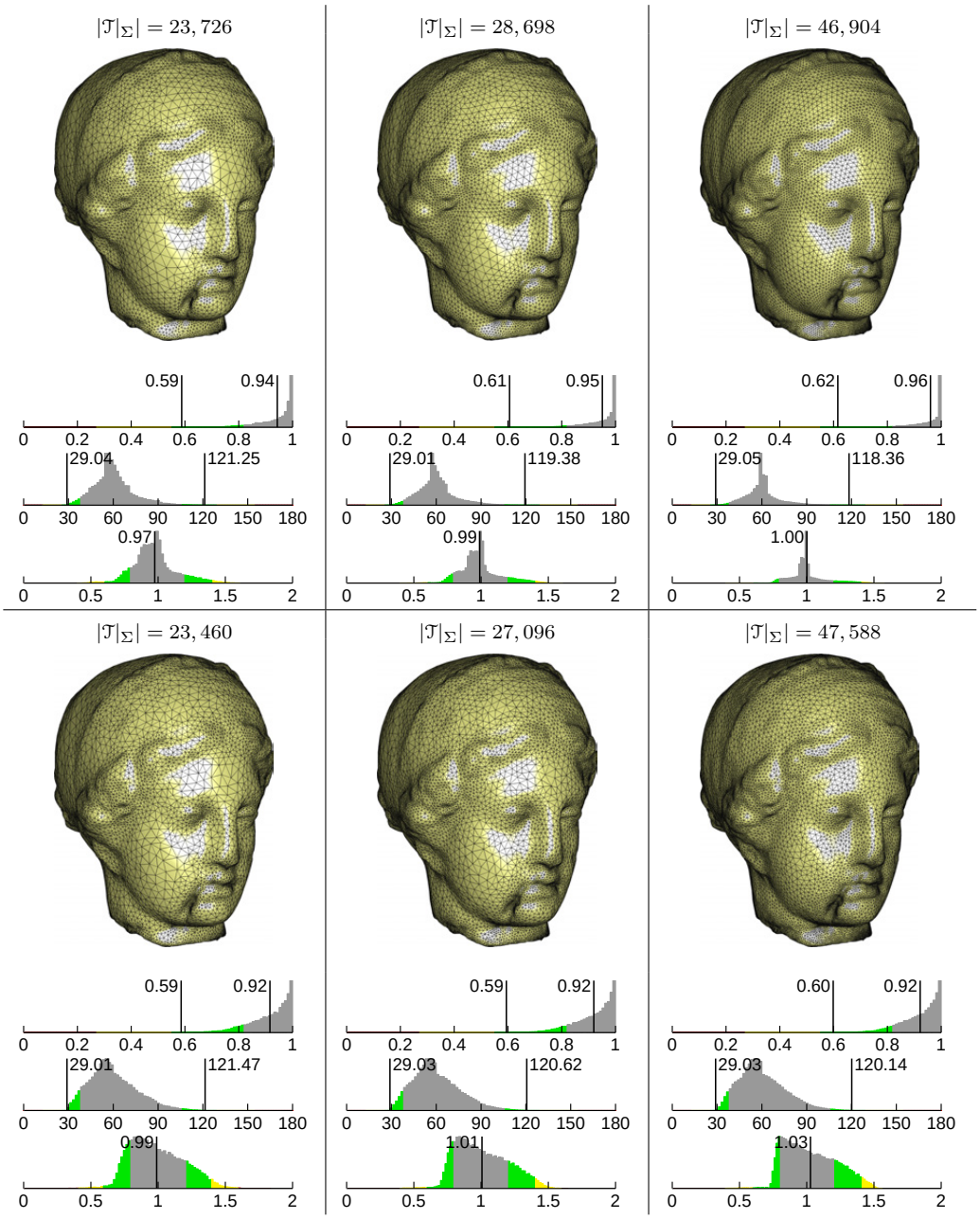


Table 4.1: Surface meshing study, comparing the performance of the Frontal-Delaunay and Delaunay-refinement algorithms on the full set of benchmark problems. Meshes are generated using medium resolution settings, such that the mesh size functions $h(\mathbf{x})$ are constructed with $g = (2/10)$. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29^\circ$. A non-uniform surface discretisation threshold is imposed, such that $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Results listed include: total element count $|\mathcal{T}|_\Sigma|$, total runtime $t(s)$, mean and minimum area-length ratios $\overline{a(f)}$, $a(f)_{\min}$ and plane angle bounds $\theta(f)_{\max}$, $\theta(f)_{\min}$.

Domain	Frontal-Delaunay						Delaunay-refinement					
	$\lvert\mathcal{T}\rvert_\Sigma\rvert$	$t(s)$	$a(f)_{\min}$	$\overline{a(f)}$	$\theta(f)_{\min}$	$\theta(f)_{\max}$	$\lvert\mathcal{T}\rvert_\Sigma\rvert$	$t(s)$	$a(f)_{\min}$	$\overline{a(f)}$	$\theta(f)_{\min}$	$\theta(f)_{\max}$
SPHERE	730	0.07	0.68	0.96	32.8°	111.3°	694	0.06	0.63	0.93	31.2°	116.7°
ELLIPSOID	1,592	0.13	0.72	0.97	32.0°	107.2°	1,488	0.11	0.63	0.93	30.2°	117.3°
FEMUR	5,392	1.35	0.64	0.96	29.3°	116.3°	5,014	1.20	0.59	0.93	29.1°	121.2°
HIP	14,728	1.44	0.59	0.95	29.0°	121.4°	14,274	1.17	0.59	0.92	29.1°	121.5°
VERTEBRA	20,426	1.87	0.60	0.94	29.0°	120.1°	19,900	1.49	0.58	0.92	29.0°	121.8°
WOOD	9,756	1.59	0.64	0.95	29.1°	116.1°	9,294	1.36	0.59	0.92	29.0°	121.5°
HAND	18,132	3.85	0.58	0.94	29.0°	121.9°	17,250	3.41	0.58	0.92	29.0°	121.7°
BLADE	26,936	3.83	0.60	0.95	29.0°	120.4°	25,538	3.16	0.58	0.92	29.0°	121.7°
ROCKER	12,600	1.49	0.60	0.95	29.1°	120.4°	11,942	1.22	0.58	0.92	29.0°	121.6°
SPIRAL 1	17,552	2.21	0.63	0.96	29.1°	116.7°	17,296	1.99	0.60	0.93	29.2°	119.7°
SPIRAL 2	22,310	2.73	0.61	0.96	29.1°	119.5°	22,306	2.47	0.60	0.93	29.1°	119.9°
BUNNY	15,026	1.81	0.59	0.95	29.0°	121.1°	14,408	1.47	0.58	0.92	29.0°	121.9°
BIMBA	32,316	8.28	0.58	0.95	29.1°	121.7°	30,514	7.16	0.58	0.92	29.0°	121.6°
VENUS	28,698	7.60	0.59	0.95	29.0°	121.1°	27,096	6.65	0.58	0.92	29.0°	121.7°
DINOSAUR	101,406	13.77	0.58	0.94	29.0°	122.0°	98,102	10.95	0.58	0.92	29.0°	121.8°

small, the element size constraints dominate.

4.6.1.2 Overall Comparisons

In addition to the detailed comparisons presented for the BUNNY and VENUS test problems, a simplified set of comparative results are also obtained for all benchmark problems, contrasting the performance of the Frontal-Delaunay and Delaunay-refinement approaches. The results of this study are presented in Table 4.1. Meshes are generated using medium resolution settings, with the mesh size function $\bar{h}(\mathbf{x})$ constructed with $g = (2/10)$. Again, tight bounds on element radius-edge ratios are specified, such that $\theta_{\min} \simeq 29^\circ$ and $\theta_{\max} \simeq 122^\circ$. A non-uniform surface discretisation threshold is enforced, setting $\bar{\epsilon} = \beta\bar{h}(\mathbf{x})$, with $\beta = (1/10)$. Analysis of the results presented in Table 4.1 confirm the trends observed in the detailed analysis carried out for the BUNNY and VENUS problems – that, given an appropriate mesh size function $\bar{h}(\mathbf{x})$, the proposed Frontal-Delaunay algorithm produces meshes that are of significantly higher shape-quality than those generated using the conventional Delaunay-refinement algorithm. Total run-times for the algorithms are also tabulated and show, firstly, that the implementations developed in this study are efficient, generating meshes containing $10,000$'s of elements in a matter of seconds. It can be seen that the Frontal-Delaunay algorithm is typically slower than the Delaunay-refinement method by a margin of approximately 10-20%. The additional computational expense is associated with the use of off-centres in the Frontal-Delaunay algorithm, specifically, the placement of size-optimal Type II vertices according to the iterative strategy described in Section 4.2. Such a procedure requires the solution of a small local system of non-linear equations, which, in the case of surface refinement, leads to the iterative computation of surface intersections. Such intersections are relatively costly, currently implemented as geometric searches over the facets of the triangulated surface Σ via traversals of a supporting AABB-tree. The development of techniques to speed-up this process has been identified as an avenue for future research.

4.6.2 User-defined Size Constraints

The performance of the surface meshing algorithms for problems involving user specified mesh size constraints $\bar{h}_u(\mathbf{x})$ is also assessed. In Figure 4.10, the results of a graded meshing study are presented, in which a set of surface meshes are generated for the BUNNY and VENUS test problems that adhere to user defined sizing constraints. The user defined constraints in these examples are illustrative-only, consisting of a simple sinusoidal variation, $\bar{h}_u(\mathbf{x}) = c_1 \sin(c_2\mathbf{z}) + c_3$, where the coefficients $c_1, c_2, c_3 \in \mathbb{R}$ are chosen based on the dimensions of the geometry and $\mathbf{z}$ is a locally aligned vertical coordinate. Note that in addition to the user specified size function $\bar{h}_u(\mathbf{x})$, a geometric size function $\bar{h}_g(\mathbf{x})$ is also constructed, with the final mesh size function $\bar{h}(\mathbf{x})$ taken as the minimum of the user-defined and geometric contributions, such that $\bar{h}(\mathbf{x}) = \min\big(\bar{h}_g(\mathbf{x}), \bar{h}_u(\mathbf{x})\big)$, as per the methods outlined in Section 4.4. The meshes generated using both the Frontal-Delaunay and Delaunay-refinement algorithms are presented in Figure 4.10. The final mesh size function $\bar{h}(\mathbf{x})$ is constructed using a Lipschitz smoothness parameter $g = (2/10)$. In all cases, a

constant radius-edge ratio is specified, equivalent to $\theta_{\min} = 29^\circ$ and $\theta_{\max} = 122^\circ$. A non-uniform surface discretisation threshold is enforced, setting $\bar{\epsilon} = \beta \bar{h}(\mathbf{x})$, with $\beta = (1/10)$. For all test problems, element quality has been catalogued, with normalised histograms of element area-length $a(f)$, plane angle $\theta(f)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean area-length measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length.

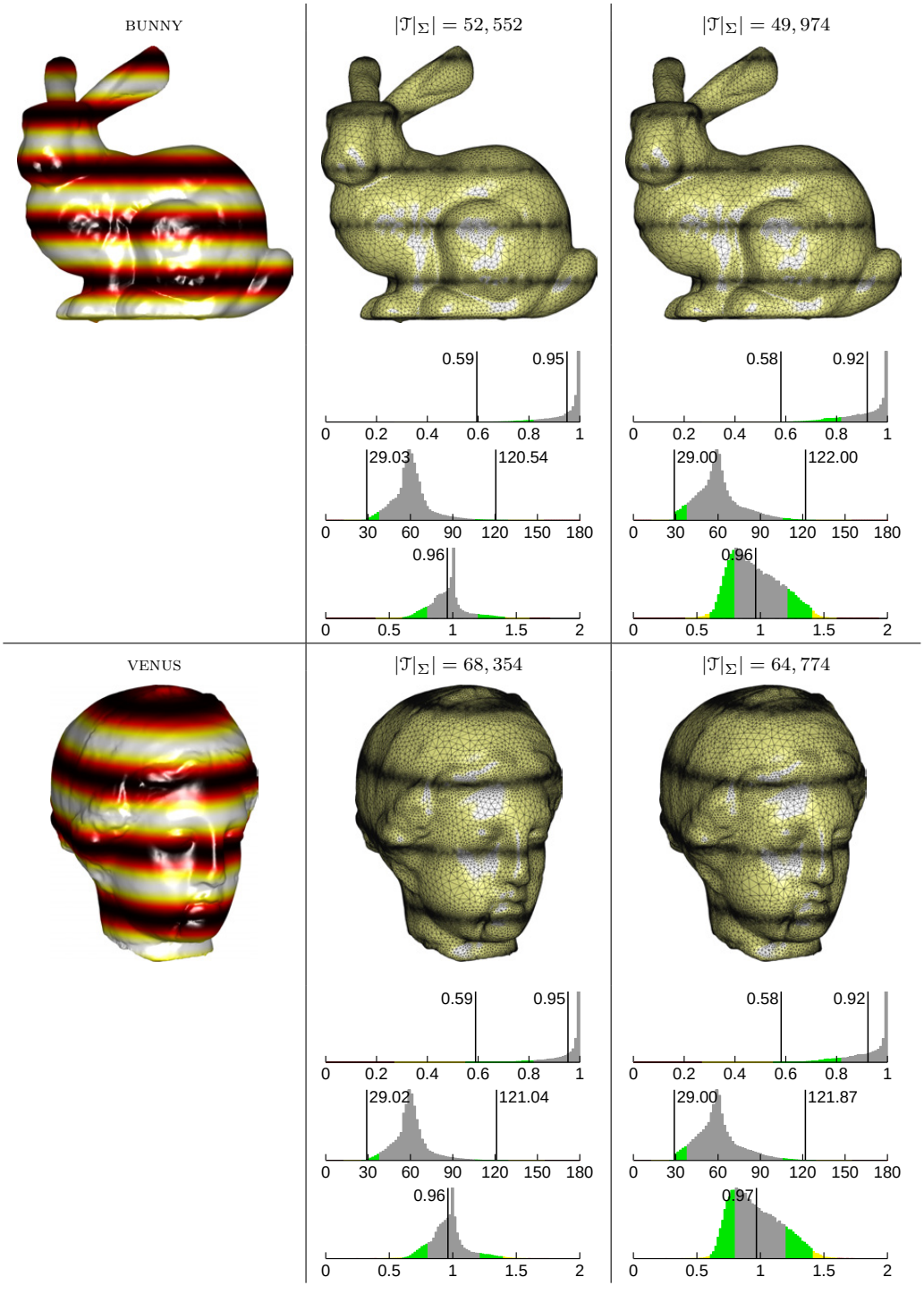
Analysis of these graded meshes confirm much of the behaviour discussed in the previous size- and shape-driven comparison studies, with the Frontal-Delaunay algorithm generating meshes of significantly higher shape- and size-quality when compared to the conventional Delaunay-refinement method. Specifically, it is noted that mean element quality is improved for both the BUNNY and VENUS benchmark problems. Note also that meshes generated using the Frontal-Delaunay technique are observed to conform more uniformly to the specified size function. Overall, these results demonstrate the effectiveness of the proposed Frontal-Delaunay method when generating high-quality graded surface meshes for user-defined sizing constraints.

4.7 Conclusions

In this chapter, I have presented a pair of algorithms designed for high-quality surface mesh generation for closed 2-manifolds embedded in $\mathbb{R}^3$. Both algorithms are based on the so-called *restricted* Delaunay tessellation, in which a surface triangulation $\text{Del}|_\Sigma(X)$, conforming to an underlying surface Σ, is constructed as a subset of a full-dimensional Delaunay tessellation $\text{Del}(X)$. The first algorithm is a 'conventional' restricted Delaunay-refinement scheme, closely modelled on the `CGALMESH` algorithm presented by Jamin et al. in [15], in which the centres of surface Delaunay balls associated with poor quality elements are inserted into the mesh as Steiner vertices. In the second algorithm, I have proposed a new restricted Frontal-Delaunay technique, generalising the ideas introduced in Chapter 3 for planar meshing algorithms to support surface domains. In this new algorithm, generalised off-centre Steiner vertices are utilised, based on a combination of size- and shape-driven refinement strategies, leading to a hybrid approach that combines many of the advantages of advancing-front and Delaunay-refinement techniques. A series of comparative experimental studies confirm the effectiveness of this new approach, demonstrating that an improvement in element shape- and size-quality is typically achieved when comparing the new Frontal-Delaunay method with conventional Delaunay-refinement techniques. Importantly, it has also been demonstrated that the new Frontal-Delaunay algorithm achieves the same theoretical optimality as the conventional Delaunay-refinement approach, satisfying constraints on element radius-edge ratios, edge length and surface discretisation thresholds. Results show that the new algorithm is an effective hybridisation of existing mesh generation techniques, combining the high element quality and mesh structure of advancing-front techniques with the theoretical guarantees of Delaunay-refinement schemes.

A number of avenues exist for future investigations – improving or generalising the Delaunay-refinement and Frontal-Delaunay surface meshing algorithms presented in this

Figure 4.10: Adaptive meshing study for the BUNNY and VENUS test cases, showing (left) contours of the user-defined mesh size functions $\bar{h}_u(\mathbf{x})$. Meshes generated using the Frontal-Delaunay (centre) and Delaunay-refinement (right) algorithms are graded to conform to these size constraints. A constant radius-edge ratio limit is specified for all cases, such that $\theta_{\min} \simeq 29°$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element area-length ratio $a(f)$, plane angle $\theta(f)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.



study. The speed of the current implementation could be further improved, by, for example, investigating alternative strategies for the placement of Type II size-optimal Steiner vertices, with the aim of minimising the number of expensive surface intersection queries required. Secondly, the applicability of the algorithms could be greatly improved by adding support for additional types of surface definitions, including, for example, support for open and non-manifold domains and alternative surface formulations, such as the parametric definitions commonly associated with Computer Aided Drawing (CAD) applications.

References

[1] Amenta, N., and Bern, M. Surface Reconstruction by Voronoi Filtering. *Discrete & Computational Geometry 22*, 4 (1999), 481–504.

[2] Amenta, N., Bern, M., and Kamvysselis, M. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 415–421.

[3] Amenta, N., Choi, S., and Kolluri, R. K. The power crust, unions of balls, and the medial axis transform. *Computational Geometry 19*, 2 (2001), 127–153.

[4] Boissonnat, J. D., and Oudot, S. Provably Good Surface Sampling and Approximation. In *ACM International Conference Proceeding Series* (2003), vol. 43, pp. 9–18.

[5] Boissonnat, J. D., and Oudot, S. Provably Good Sampling and Meshing of Surfaces. *Graphical Models 67*, 5 (2005), 405–451.

[6] Boissonnat, J. D., and Oudot, S. Provably Good Sampling and Meshing of Surfaces. *Graphical Models 67*, 5 (2005), 405 – 451. ¡ce:title¿Solid Modeling and Applications¡/ce:title¿.

[7] Boissonnat, J. D., and Oudot, S. Provably Good Sampling and Meshing of Lipschitz Surfaces. In *Proceedings of the twenty-second annual symposium on Computational geometry* (New York, NY, USA, 2006), SCG '06, ACM, pp. 337–346.

[8] Cheng, S. W., Dey, T. K., and Ramos, E. A. Delaunay Refinement for Piecewise Smooth Complexes. *Discrete & Computational Geometry 43*, 1 (2010), 121–166.

[9] Cheng, S. W., Dey, T. K., and Shewchuk, J. R. *Delauay Mesh Generation*. Taylor & Francis, New York, 2013.

[10] Dey, T. K., and Zhao, W. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica 38*, 1 (2004), 179–200.

[11] Engwirda, D., and Ivers, D. Face-centred Voronoi Refinement for Surface Mesh Generation. *Procedia Engineering 82* (2014), 8–20.

[12] Engwirda, D., and Ivers, D. Size-optimal Steiner points for Delaunay-refinement on curved surfaces. *arXiv preprint arXiv:1501.04002* (2015).

[13] Erten, H., and Üngör, A. Quality Triangulations with Locally Optimal Steiner Points. *SIAM J. Sci. Comp. 31*, 3 (2009), 2103–2130.

[14] Frey, P. J., Borouchaki, H., and George, P. L. 3D Delaunay Mesh Generation Coupled with an Advancing-Front Approach. *Computer Methods in Applied Mechanics and Engineering 157*, 12 (1998), 115 – 131.

[15] Jamin, C., Alliez, P., Yvinec, M., and Boissonnat, J. D. CGALmesh: A Generic Framework for Delaunay Mesh Generation. Tech. rep., INRIA, 2013.

[16] Mavriplis, D. J. An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *Journal of Computational Physics 117*, 1 (1995), 90 – 101.

[17] Persson, P. O. *Mesh Generation for Implicit Geometries*. PhD thesis, Boston, Massachusetts, 2004.

[18] Persson, P.-O. Mesh Size Functions for Implicit Geometries and PDE-based Gradient Limiting. *Engineering with Computers 22*, 2 (2006), 95–109.

[19] Rebay, S. Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal of Computational Physics 106*, 1 (1993), 125 – 138.

[20] Remacle, J. F., Henrotte, F., Carrier-Baudouin, T., Béchet, E., Marchandise, E., Geuzaine, C., and Mouton, T. A Frontal-Delaunay Quad-Mesh Generator using the $\mathbb{L}^\infty$-norm. *International Journal for Numerical Methods in Engineering 94*, 5 (2013), 494–512.

[21] Rypl, D. *Sequential and Parallel Generation of Unstructured 3D Meshes.* PhD thesis, CTU in Prague, 1998.

[22] Rypl, D. Approaches to Discretization of 3D Surfaces. In *CTU Reports*, vol. 7. CTU Publishing House, Prague, Czech Republic, 2003.

[23] Schreiner, J., Scheiclegger, C., and Silva, C. High-Quality Extraction of Isosurfaces from Regular and Irregular Grids. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (2006), 1205–1212.

[24] Schreiner, J., Scheidegger, C. E., Fleishman, S., and Silva, C. T. Direct (Re)Meshing for Efficient Surface Processing. *Computer Graphics Forum 25*, 3 (2006), 527–536.

[25] Üngör, A. Off-centers: A New Type of Steiner Points for Computing Size-optimal Guaranteed-quality Delaunay Triangulations. *Computational Geometry: Theory and Applications 42*, 2 (2009), 109–118.

Chapter 5

Volumetric Mesh Generation

In this chapter, I present a Frontal-Delaunay meshing algorithm for volumetric domains in $\mathbb{R}^3$. This new algorithm is an extension of the Frontal-Delaunay methods developed for planar and surface domains in Chapters 3 and 4, and is designed to combine the high-quality results achieved using advancing-front techniques with the provable bounds and theoretical guarantees of Delaunay-refinement schemes. Like the surface meshing techniques presented in the previous chapter, the algorithms for volumetric mesh generation are based on the *restricted* Delaunay triangulation – a subset of the full-dimensional Delaunay tessellation conforming to the domain of interest. I review both the mechanics and theoretical development of existing restricted Delaunay-refinement algorithms before introducing the new strategy. Exploiting ideas similar to those presented in Chapters 3 and 4, I show that the use of 'off-centre' Steiner vertices, positioned along edges in the associated Voronoi complex, typically leads to an improvement in both the shape- and size-quality of the resulting tessellations. Specifically, I show that in addition to conventional shape-driven refinement methods, based on element radius-edge ratios, a new size-optimal refinement strategy can be realised by positioning off-centre vertices such that a local mesh size function is satisfied. The use of this sizing function to generate graded meshes adhering to user defined size constraints is also explored.

Three-dimensional mesh generation is typically plagued by the existence of low-quality *sliver* elements, and I review the conditions under which these type of elements are generated within the Delaunay framework. Furthermore, I investigate the idea of topological optimality for meshes in $\mathbb{R}^3$, and demonstrate, using a set of simple examples, that the topology of the Delaunay tessellation is non-optimal in terms of element dihedral angles and/or volume-length ratios. Despite these theoretical shortcomings, I present a simple modification to the Delaunay-refinement and Frontal-Delaunay algorithms that leads to the elimination of elements with pathologically poor dihedral angles. This strategy is an extension of the ideas presented by Cheng, Dey and Shewchuk in [5], in which element refinement is driven not by the conventional radius-edge ratios, but instead using alternative measures of element shape-quality. I present a new refinement strategy based on element *volume-length* ratios, in which elements of small volume-length ratios are marked for refinement. Such methods are shown to produce meshes without unattractively small

or large dihedral angles, at the expense of the introducing of additional Steiner vertices. I investigate the performance of these new methods experimentally, and undertake a series of comparative studies, contrasting the performance of the new algorithms with typical Delaunay-refinement techniques. Experiments are conducted using a range of complex benchmarks, verifying the robustness and practical performance of the proposed schemes. Work presented in this chapter appears in [7].

5.1 Restricted Delaunay Refinement

Delaunay-refinement algorithms for volumetric meshing operate by incrementally introducing new Steiner vertices into an initially coarse Delaunay triangulation of the volume to be meshed. Such triangulations typically incorporate a high-quality *embedded* triangulation of the bounding surface of the domain. Consistent with the techniques presented in Chapter 4 for surface meshing, volumetric refinement schemes are designed not only to ensure that the resulting mesh satisfies element shape and size constraints, but that the geometry and topology of the mesh itself is an accurate piecewise approximation to both the volumetric domain and its bounding surface. The volumetric meshing algorithms presented in this chapter are based on the so-called *restricted* Delaunay surface and volumetric tessellations $\text{Del}\,|_{\Sigma}(X)$ and $\text{Del}\,|_{\Omega}(X)$ – triangular and tetrahedral subcomplexes of the full-dimensional Delaunay tessellation $\text{Del}(X)$. Specifically, the the embedded restricted surface triangulation $\text{Del}\,|_{\Sigma}(X)$ contains the 2-faces $f \in \text{Del}(X)$ that *best* approximate the bounding surface Σ, while the restricted volumetric triangulation $\text{Del}\,|_{\Omega}(X)$ contains the 3-simplexes internal to the volumetric domain Ω. The restricted surface and volumetric Delaunay tessellations are introduced and defined in Section 2.2, where additional theoretical discussion is presented. Additionally, an efficient incremental algorithm for the construction and maintenance of such structures is detailed in Section 4.1.

5.1.1 An Existing Algorithm

The development of provably-good Delaunay-refinement schemes for the meshing of volumetric domains is an ongoing area of research. In this section, I present an algorithm for the meshing of volumes enclosed by smooth 2-manifolds embedded in $\mathbb{R}^3$, that is adapted largely from the methods presented by Oudot, Rineaua and Yvinec in [18]. This method is largely equivalent to the `CGALMESH` algorithm, available as part of the `CGAL` package, and summarised by Jamin, Alliez, Yvinec and Boissonnat in [12]. A similar algorithm is also outlined by Cheng, Dey and Shewchuk in [5]. I will refer to the algorithm presented in this section as the 'conventional' Delaunay-refinement approach, due to its direct use of circumcentre-based Steiner vertices. Consistent with the surface meshing techniques developed in Chapter 4, I restrict my attention to so-called 're-meshing' algorithms in this study, in which the underlying surface definition Σ is specified as an existing manifold triangular complex $\mathcal{P}$. See Section 4.1 for a discussion of the scope and appropriateness of such a restriction.

Following the approach described by Jamin et al. in [12], the Delaunay-refinement algorithm takes as input a volumetric domain, described by a closed 2-manifold $\Sigma \subseteq \mathbb{R}^3$, an upper bound on the allowable element radius-edge ratio $\bar{\rho}$, a mesh size function $\bar{h}(\mathbf{x})$ defined at all points enclosed by the surface Σ and an upper bound on the allowable surface discretisation error $\bar{\epsilon}(\mathbf{x})$. The input surface Σ encloses a bounded volume Ω. The algorithm returns a triangulation $\mathcal{T}|_\Sigma$ of the surface Σ, where $\mathcal{T}|_\Sigma$ is a restricted Delaunay surface triangulation of a point-wise sampling $X \in \Sigma$, such that $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$. Additionally, the algorithm also returns a triangulation $\mathcal{T}|_\Omega$ of the enclosed volume Ω, where $\mathcal{T}|_\Omega$ is a restricted Delaunay volumetric triangulation $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$. Both $\text{Del}\,|_\Sigma(X)$ and $\text{Del}\,|_\Omega(X)$ are sub-complexes of the full-dimensional Delaunay tessellation $\text{Del}(X)$. Note that $\text{Del}\,|_\Sigma(X)$ is a triangular complex, while $\text{Del}\,|_\Omega(X)$ and $\text{Del}(X)$ are tetrahedral complexes. The Delaunay-refinement algorithm is summarised in Algorithm 5.1.1.

The Delaunay-refinement algorithm guarantees, firstly, that all elements in the output volume triangulation $\tau \in \mathcal{T}|_\Omega$ satisfy constraints on both the element shape and size, such that $\rho(\tau) \leq \bar{\rho}$, and $h(\tau) \leq \bar{h}(\mathbf{x}_\tau)$. Secondly, all elements in the embedded surface triangulation $f \in \mathcal{T}|_\Sigma$ satisfy similar element shape and size constraints as well as an upper bound on the allowable surface discretisation error, ensuring that $\epsilon(f) \leq \bar{\epsilon}(\mathbf{x}_f)$. Finally, a consistency constraint is also enforced, ensuring that the restricted surface triangulation is truly embedded within the volumetric tessellation, such that $\text{Del}\,|_\Sigma(X) = \partial\,\text{Del}\,|_\Omega(X)$, where $\partial\,\text{Del}\,|_\Omega(X)$ is the set of boundary 2-faces for the tetrahedral complex $\text{Del}\,|_\Omega(X)$.

Compared with the restricted surface meshing algorithm outlined in Chapter 4, it is only this last consistency constraint that is new. Such a condition ensures that no elements in the restricted surface triangulation $\text{Del}\,|_\Sigma(X)$ are *orphaned* – spanning locally under-resolved regions of the surface Σ that do not enclose elements in the associated volumetric tessellation $\text{Del}\,|_\Omega(X)$. In such configurations, a surface facet $f \in \text{Del}\,|_\Sigma(X)$ is *not* face-adjacent to any tetrahedra $\tau_i, \tau_j \in \text{Del}\,|_\Omega(X)$[1], and, as a result, $f \notin \partial\,\text{Del}\,|_\Omega(X)$. Such issues are readily addressed through additional refinement – increasing local resolution on the surface Σ through a subdivision of the surface facet f. See [5] for extended discussions on this topic.

Based on the properties of the coarse initial volume triangulation $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$ and the embedded surface triangulation $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$, generated using the restricted Delaunay surface meshing algorithms presented in Chapter 4, it is known that the triangulations $\mathcal{T}|_\Sigma$ and $\mathcal{T}|_\Omega$ are good piecewise linear approximations to the bounding surface Σ and volume Ω, provided that the magnitude of the mesh size function $\bar{h}(\mathbf{x})$ is sufficiently small, as per the discussions detailed in Chapter 4. Under such conditions it is known that the triangulations $\mathcal{T}|_\Sigma$ and $\mathcal{T}|_\Omega$ are homeomorphic to the underlying surface and volume definitions Σ and Ω, and that the geometric properties of $\mathcal{T}|_\Sigma$ and $\mathcal{T}|_\Omega$ converge toward the true normals, curvature, area and volume of the surface Σ and volume Ω as $\bar{h}(\mathbf{x}) \to 0$.

[1] The surface facet $f \in \text{Del}\,|_\Sigma(X)$ is clearly face-adjacent to a tetrahedral pair $\tau_i, \tau_j \in \text{Del}(X)$, but, in such cases, neither τ_i or τ_j are elements of the restricted volumetric tessellation $\text{Del}\,|_\Omega(X)$, leaving the surface facet f orphaned.

Algorithm 5.1.1 Restricted Delaunay Volume Refinement

function DelaunayVolume($\Sigma, \Omega, \bar{\rho}, \bar{\epsilon}(\mathbf{x}), \bar{h}(\mathbf{x}), \mathcal{T}|_\Sigma, \mathcal{T}|_\Omega$)
  Call DelaunaySurface($\Sigma, \Omega, \bar{\rho}, \bar{\epsilon}(\mathbf{x}), \bar{h}(\mathbf{x}), \mathcal{T}|_\Sigma, \mathcal{T}|_\Omega$) – generating a high-quality surface triangulation $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$ for a point-wise surface sample $X \in \Sigma$ and initialising a coarse volumetric triangulation $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$.
  Enqueue all 2- and 3-simplexes $Q|_\Sigma \leftarrow f \in \text{Del}\,|_\Sigma(X)$ and $Q|_\Omega \leftarrow \tau \in \text{Del}\,|_\Omega(X)$. Simplexes are enqueued if BadSimplex2(f) or BadSimplex3(τ) returns true.
  while ($Q|_\Sigma \neq \emptyset$ or $Q|_\Omega \neq \emptyset$) **do** ▷ {main refinement loop}
    if ($Q|_\Sigma \neq \emptyset$) **then**
      Call RefineSimplex2($f \leftarrow Q|_\Sigma$)
    else
      Call RefineSimplex3($\tau \leftarrow Q|_\Omega$)
    end if
    for all (updated $\tau \in \text{Del}(X)$) **do**
      Call RestrictedDT($\tau, \text{Del}\,|_\Sigma(X), \text{Del}\,|_\Omega(X)$)
    end for
    Update $Q|_\Sigma$ & $Q|_\Omega$ to reflect changes in $\text{Del}\,|_\Sigma(X)$ & $\text{Del}\,|_\Omega(X)$.
  end while
  return $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$ and $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$
end function

function RefineSimplex2(f) ▷ {surface refinement}
  Call SurfaceDelaunayBall($f, \text{B}(\mathbf{c}, r)_{\max}$)
  Insert Steiner point $X \leftarrow \mathbf{c}_{\max}$ and update $\text{Del}(X) \leftarrow X$.
end function

function RefineSimplex3(τ) ▷ {volume refinement}
  Form the Steiner point $\mathbf{c}$ for the simplex τ.
  if (c encroaches on any surface facet $f \in \text{Del}\,|_\Sigma(X)$) **then**
    Call RefineSimplex2(f)
  else
    Insert Steiner point $X \leftarrow \mathbf{c}$ and update $\text{Del}(X) \leftarrow X$.
  end if
end function

function BadSimplex2(f) ▷ {termination criteria}
  return ($\rho(f) > \bar{\rho}$) or ($\epsilon(f) > \bar{\epsilon}(\mathbf{x}_f)$) or ($h(f) > \bar{h}(\mathbf{x}_f)$) or $f \notin \partial\mathcal{T}|_\Omega$
end function

function BadSimplex3(τ) ▷ {termination criteria}
  return ($\rho(\tau) > \bar{\rho}$) or ($h(\tau) > \bar{h}(\mathbf{x}_\tau)$)
end function

The Delaunay-refinement algorithm begins by constructing an initial surface triangulation $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$, based on a well-distributed point-wise sampling $X \in \Sigma$ via a call to the restricted surface Delaunay-refinement meshing algorithm presented in Chapter 4. In this study, for reasons of consistency, the volumetric Delaunay-refinement algorithm is combined with the surface Delaunay-refinement algorithm only, while the surface Frontal-Delaunay algorithm is used only in combination with the volumetric Frontal-Delaunay algorithm presented in subsequent sections. This is segregation is somewhat arbitrary, and use of the various combinations of Delaunay-refinement and Frontal-Delaunay surface and volumetric meshing algorithms is easily achieved in practice. Such investigations are not pursued in the present study. Additionally, again for reasons of consistency, a common mesh size function $\bar{h}(\mathbf{x})$ is used for both the surface and volumetric meshing phases in all cases.

Consistent with the methods outlined in previous chapters, the full-dimensional tessel-

lation $\mathrm{Del}(X)$ is maintained incrementally, using the efficient Bowyer-Watson framework outlined in Chapter 2. The restricted surface and volumetric triangulations, $\mathrm{Del}\,|_\Sigma(X)$ and $\mathrm{Del}\,|_\Omega(X)$, are derived from $\mathrm{Del}(X)$ using the incremental restricted Delaunay algorithms presented in Chapter 4, in which the intersection and enclosure of edges and vertices in the associated Voronoi diagram $\mathrm{Vor}(X)$ and the bounding surface Σ are computed explicitly.

The main loop of the volumetric meshing algorithm proceeds to incrementally refine any 2-faces $f \in \mathrm{Del}\,|_\Sigma(X)$ or 3-simplexes $\tau \in \mathrm{Del}\,|_\Omega(X)$ that do not satisfy either the radius-edge, element size, surface discretisation or topological constraints, with triangles $f \in \mathrm{Del}\,|_\Sigma(X)$ refined in preference to tetrahedra $\tau \in \mathrm{Del}\,|_\Omega(X)$. Furthermore, all elements are ordered according to their radius-edge ratios $\rho(f)$ and $\rho(\tau)$, ensuring that elements with the largest radius-edge ratio are refined at each iteration. Individual elements are refined based on their circumballs, with a triangle $f \in \mathrm{Del}(X)$ eliminated by inserting the centre of the largest surface Delaunay ball $\mathrm{SDB}(f)$ into the tessellation $\mathrm{Del}(X)$, while a tetrahedron $\tau \in \mathrm{Del}\,|_\Omega(X)$ is eliminated by inserting the centre of its diametric ball into the existing tessellation. Additionally, the *encroachment* of surface facets is managed, with the insertion of a volumetric Steiner vertex deferred if the new point lies inside the surface Delaunay ball associated with any surface facet $f \in \mathrm{Del}\,|_\Sigma(X)$. In such cases, the surface facet is instead refined, ensuring that the local resolution of the embedded surface triangulation reflects that of the neighbouring volumetric tessellation. These processes are a direct generalisation of the circumcentre-based insertion methods introduced in Ruppert's algorithm for planar domains, discussed in Chapter 3.

As a consequence of changes to the full-dimensional tessellation $\mathrm{Del}(X)$ following the insertion of a new Steiner vertex, corresponding updates to the restricted triangulations $\mathrm{Del}\,|_\Sigma(X)$ and $\mathrm{Del}\,|_\Omega(X)$ are instigated, ensuring that all tessellation objects remain in-sync throughout the refinement process. The Delaunay-refinement algorithm terminates when all 2-faces $f \in \mathrm{Del}\,|_\Sigma(X)$ and all 3-simplexes $\tau \in \mathrm{Del}\,|_\Omega(X)$ satisfy all associated constraints. In the case of surface facets, this ensures that all radius-edge, element size, surface discretisation and topological constraints are satisfied. Considering the volumetric elements, a set of consistent constraints are satisfied, ensuring that[1] $\rho(\tau) \leq \bar{\rho}$ and $h(\tau) \leq \alpha \bar{h}(\mathbf{x}_\tau)$, respectively, where the element size $h(\tau)$ is proportional is the radius of the associated circumball $\mathrm{B}(\mathbf{c}, r)$, such that[2] $h(\tau) = (4/\sqrt{6})r$, and $\bar{h}(\mathbf{x}_\tau)$ is sampled at the centre of the associated diametric ball.

5.1.2 Discussion

Like the surface meshing algorithms presented in Chapter 4, the performance of the volumetric Delaunay-refinement scheme can be understood in terms of both geometrical and topological fidelity and element shape- and size-quality. Recalling the results of

[1] The scalar $\alpha = \frac{4}{3}$, to ensure that the mean element size does not, on average, undershoot the target size $\bar{h}(\mathbf{x}_\tau)$, as per the discussions outlined in Section 3.2.

[2] The coefficient $\sqrt{6}/4$ represents the mapping between the edge length and diametric ball radius for an equilateral element. Such scaling ensures that size constraints are applied with respect to mean edge length.

Boissonnat and Oudot [1, 2] presented in Proposition 4.1, the surface and volumetric tessellations $\mathcal{T}|_\Sigma = \mathrm{Del}\,|_\Sigma(X)$ and $\mathcal{T}|_\Omega = \mathrm{Del}\,|_\Omega(X)$ are guaranteed to be 'good' topological and geometrical approximations to the underlying domain Ω when the point-wise sampling $X \in \Omega$ is a so-called *'loose' γ-sample*. Such behaviour is assured when the mesh size function is sufficiently small, such that $\bar{h}(\mathbf{x}) \leq \gamma d_{\mathrm{M}}(\mathbf{x})$, where $d_{\mathrm{M}}(\mathbf{x})$ is the minimum distance to the medial axis of Ω and $\gamma \leq 0.08$. These issues are discussed in detail in Section 4.1. The development of detailed theoretical analysis for volumetric meshing algorithms is an evolving area of research, and detailed remarks concerning the performance of Algorithm 5.1.1 are omitted in this study. Cheng, Dey and Shewchuk [5] show that termination and convergence can be guaranteed for a similar restricted Delaunay-refinement algorithm, given that $\bar{\rho} \geq 2$.

5.2 Restricted Frontal-Delaunay Methods

Frontal-Delaunay algorithms are a hybridisation of advancing-front and Delaunay-refinement techniques, in which a Delaunay triangulation is used to define the topology of a mesh while new Steiner vertices are inserted in a manner consistent with advancing-front methodologies. In practice, such techniques have been observed to produce very high-quality meshes, inheriting the smooth, semi-structured vertex placement of pure advancing-front methods and the optimal mesh topology of Delaunay-based approaches. Frontal-Delaunay methods have previously been for volumetric meshing studies, with early work by Rebay [19] and Mavriplis [16, 17] incorporating extensions of their original two-dimensional algorithms to simple three-dimensional configurations, while Frey, Borouchaki, and George present a fully three-dimensional method in [10]. While these methods differ slightly in the methodologies used to introduce new Steiner vertices, all involve the size-driven placement of new points about existing 'frontal' faces. Specifically, the techniques of Rebay involve a direct generalisation of his two-dimensional algorithm discussed in Chapter 3, whereby new Steiner vertices are positioned along edges in the associated Voronoi diagram such that a prescribed mesh size function is satisfied. These existing methods do not typically incorporate element shape-based refinement strategies – an important differentiator for the new Frontal-Delaunay technique developed in this thesis, where the use of both shape- and size driven refinement schemes are investigated. Such an approach is consistent with the algorithms developed for planar and surface meshing problems in Chapters 3 and 4.

5.2.1 Off-centres

The new Frontal-Delaunay algorithm presented here is based on a generalisation of the 'off-centre' techniques introduced in Chapters 3 and 4 for planar and surface mesh generation, in which new Steiner vertices are positioned along edges in the associated Voronoi complex. Such an approach shares some similarties with the *generalised* refinement strategies investigated recently by by Chernikov, Chrisochoides and Foteinos in [6, 8]. In contrast to the strategies developed previously for the planar and surface cases, I pursue

the development of a size-optimal scheme only in the volumetric case. This decision was taken in response to the small number of shape-optimal vertices that are typically selected by the planar and surface-based Frontal-Delaunay algorithms in practice. Nonetheless, further investigation of shape-optimal point-placement strategies, based on a generalisation of ideas presented in previous chapters, is an interesting avenue for future research.

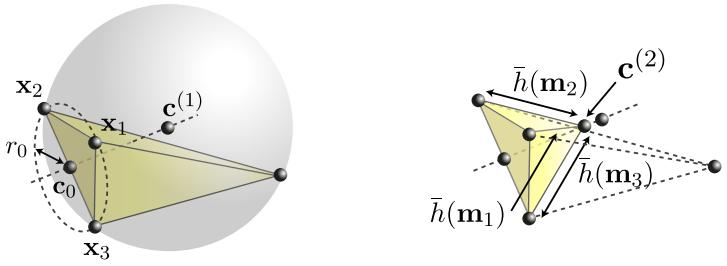
In this study, I consider the use of off-centre Steiner vertices to simulate the vertex placement strategy of a conventional advancing-front approach, while also preserving the framework of a Delaunay-refinement meshing algorithm. The aim of such a strategy is to recover the high element qualities and smooth, semi-structured meshes generated by frontal methods in practice, while inheriting the guaranteed bounds of Delaunay-refinement based methods. Advancing-front algorithms typically incorporate a mesh size function $\bar{h}(\mathbf{x})$, a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}^+$ defined over the domain to be meshed, where $\bar{h}(\mathbf{x})$ represents the desired edge length $\|\mathbf{e}\|$ at any point $\mathbf{x} \in \Omega$. This mesh size function typically incorporates size constraints dictated by the both the user and the geometry of the domain to be meshed. Since both surface and volumetric meshing problems are embedded in $\mathbb{R}^3$, a common mesh size function is used in both cases. I refer the reader to Section 4.4 for additional discussions of the methods used to construct size functions appropriate for surface and volumetric mesh generation.

The proposed Frontal-Delaunay algorithm developed in this study is an extension of the restricted Delaunay-refinement algorithm presented in Section 5.1, modified to use off-centre rather than circumcentre-based refinement schemes. The basic framework of the algorithm is consistent with the Delaunay-refinement algorithm described previously, in which an initially coarse tessellation of a bounded volume Ω, obtained as output from the Frontal-Delaunay surface meshing algorithm described in Chapter 4, is refined through the introduction of additional Steiner vertices $X \in \Omega$ until all element shape, size, surface error and topological constraints are satisfied. Following the approach introduced by Ruppert and Shewchuck, any encroached 2-faces $f \in \mathrm{Del}\,|_{\Sigma}(X)$ are also refined, preserving the integrity of the embedded surface triangulation $\mathcal{T}|_{\Sigma} = \mathrm{Del}\,|_{\Sigma}(X)$. A conforming tetrahedral complex $\mathcal{T}|_{\Omega} = \mathrm{Del}\,|_{\Omega}(X)$ is constructed through consideration of the set of restricted 3-simplexes of the full-dimensional tessellation $\mathrm{Del}(X)$. The constraints satisfied by the Frontal-Delaunay algorithm are identical to those introduced previously for the restricted Delaunay-refinement scheme, with upper bounds on the radius-edge ratio $\bar{\rho}$ and element size $\bar{h}(\mathbf{x}_{\tau})$ all required to be satisfied for convergence. Furthermore, it is required that similar constraints, including bounds on the surface discretisation error $\bar{\epsilon}(\mathbf{x})$ and local topology are satisfied by the embedded surface triangulation $\mathcal{T}|_{\Sigma} = \mathrm{Del}\,|_{\Sigma}(X)$, ensuring that the surface triangulation is an adequate representation of the bounding surface Σ, and is consistent with the boundary of the tetrahedral complex $\partial\,\mathrm{Del}\,|_{\Omega}(X)$, as outlined in Chapter 4. See Algorithm 5.1.1 for a detailed summary of the method.

5.2.2 Point-placement Strategy

The Steiner vertex introduced when refining a 3-simplex $\tau \in \mathrm{Del}\,|_{\Omega}(X)$ is an off-centre, constructed based on local adherence to element size and shape constraints. The off-

Figure 5.1: Off-centre constructions for a 3-simplex $\tau \in \mathrm{Del}\,|_\Omega(X)$, illustrating (i) the local edge of the Voronoi diagram associated with the small face $f_0 \in \tau$, and (ii) placement of the size-optimal vertex $\mathbf{c}^{(2)}$ such that local size constraints $\bar{h}(\mathbf{x}_\tau)$ are enforced.



(i) (ii)

centres introduced in this study involve the placement of two distinct kinds of Steiner vertices. Type I vertices, $\mathbf{c}^{(1)}$, are equivalent to conventional element circumcentres, and are used to satisfy constraints on the element radius-edge ratios. Type II vertices, $\mathbf{c}^{(2)}$, are *size-optimal* points, designed to satisfy element sizing constraints in a locally optimal fashion. Adopting the *generalised* off-centre framework presented in Chapters 3 and Chapter 4, the 'ideal' location of the size-optimal off-centre $\mathbf{c}^{(2)}$ is based on a consideration of the isosceles tetrahedron σ formed about the *small* 2-face $f_0 \in \tau$, where f_0 is chosen as the face with the smallest associated circumradius. The point $\mathbf{c}^{(2)}$ is positioned to ensure that σ satisfies local size constraints. As mentioned previously, the use of shape-optimal Type III vertices is not explored in the present study in the context of volumetric meshing.

Given a *refinable* 3-simplex $\tau \in \mathrm{Del}\,|_\Omega(X)$, the size-optimal Type II vertex $\mathbf{c}^{(2)}$ is placed following a generalisation of the approach introduced by Rebay. The point $\mathbf{c}^{(2)}$ is positioned along the Voronoi segment $\mathbf{v}_f \in \mathrm{Vor}(X)$ associated with the small 2-face $f_0 \in \tau$, such that the size of the new tetrahedron $h(\sigma)$ satisfies local constraints. Specifically, the altitude of the new element is calculated to ensure that the three new edges of σ are not too long, such that $\|\mathbf{e}_i\| \simeq h(\mathbf{m}_i)$, where the $\mathbf{m}_i$'s are the edge midpoints for $i = 1, 2, 3$. Given that $\mathrm{Del}\,|_\Omega(X)$ is Delaunay, a number of important properties regarding the face-adjacent Voronoi segment $\mathbf{v}_f$ are known, including, firstly, that $\mathbf{v}_f$ is orthogonal to the associated 2-face f_0, and, secondly, that $\mathbf{v}_f$ passes through the centre of the diametric ball $\mathrm{B}(\mathbf{c}_0, r_0)$ associated with the 2-face f_0. Considering the right-triangle formed from an existing vertex $\mathbf{x}_i \in \tau$, the centre $\mathbf{c}_0$ and the size-optimal point $\mathbf{c}^{(2)}$, the element altitude $a^{(2)}$ can be expressed in terms of each of the edge-based size constraints

$$a_i^{(2)} = \left(\bar{h}(\mathbf{m}_i)^2 - r_0^2\right)^{\frac{1}{2}} \tag{5.1}$$

where $\bar{h}(\mathbf{m}_i)$ is the desired edge length. Given the set of altitudes, the position of the size-optimal point $\mathbf{c}^{(2)}$ can be expressed as

$$\mathbf{c}^{(2)} = \mathbf{c}_0 + \tfrac{1}{3}\left(a_1^{(2)} + a_2^{(2)} + a_3^{(2)}\right)\hat{\mathbf{v}} \tag{5.2}$$

where $\mathbf{c}_0$ is the circumcentre associated with the 2-face f_0 and $\hat{\mathbf{v}}$ is the *frontal* unit direction vector associated with the Voronoi edge $\mathbf{v}_f$. Note that, for non-uniform $\bar{h}(\mathbf{x})$, this expression is non-linear, with the altitudes $a_i{}^{(2)}$ depending on the evaluation of the mesh size function at the edge midpoints $\bar{h}(\mathbf{m}_i)$ and visa-versa. In practice, since the mesh size function $\bar{h}(\mathbf{x})$ is known to be Lipschitz smooth, a simple iterative predictor-corrector procedure is sufficient to resolve these expressions approximately.

Using the size-optimal Type II point $\mathbf{c}^{(2)}$ and the Type I point $\mathbf{c}^{(1)}$, the final position of the refinement point $\mathbf{c}$ for the facet f is calculated. The point $\mathbf{c}$ is selected to satisfy the *limiting* local constraints, setting

$$\mathbf{c} = \begin{cases} \mathbf{c}^{(2)}, & \text{if } \left(d^{(2)} \leq d^{(1)}\right) \text{ and } \left(d^{(2)} \geq r_0\right), \\ \mathbf{c}^{(1)}, & \text{otherwise} \end{cases} \tag{5.3}$$

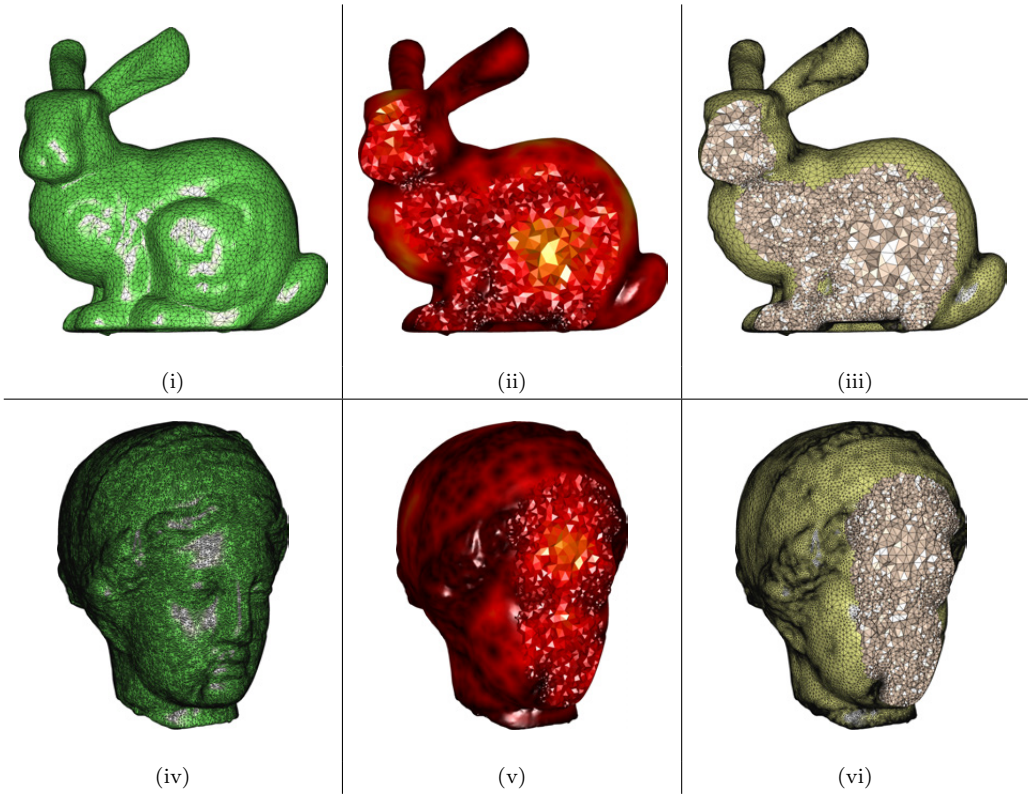
where the $d^{(i)} = \|\mathbf{c}^{(i)} - \mathbf{c}_0\|$ are distances from the centre of the frontal face f_0 to the Type I and Type II vertices, respectively. The cascading selection criteria is designed to ensure that the refinement scheme smoothly degenerates to that of a conventional circumcentre-based Delaunay-refinement strategy in limiting cases, while using locally shape-optimal points where possible. Specifically, the condition $d^{(2)} \leq d^{(1)}$ guarantees that $\mathbf{c}$ lies no further from the frontal face f_0 than the centre of the circumball of τ. Additionally, the condition $d^{(2)} \geq r_0$ ensures that the diametric ball of the frontal face f_0 remains empty. Such behaviour guarantees that the size-optimal scheme is only selected when f_0 is sufficiently small with respect to the local mesh size function – ensuring that f_0 is a good *frontal* face candidate in the context of a conventional advancing-front scheme. A shape-based strategy, guaranteed to reduce element radius-edge ratios, is selected otherwise.

5.3 Mesh Size Functions

The construction of high-quality mesh size functions is an important aspect of the Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 5.1 and 5.2. Recalling that in both cases, an initial surface triangulation is first formed via calls to the surface meshing algorithms presented in Chapter 4, a consistent mesh size function is used for both the surface and volumetric meshing phases, ensuring that compatible surface and volumetric mesh resolution is achieved. As such, the mesh size function $\bar{h}(\mathbf{x})$ used in this study is a piecewise linear g-Lipschitz function $f : \mathbb{R}^3 \to \mathbb{R}^+$, defined on a supporting tetrahedral complex $\mathcal{S} = \text{Del}(Y)$ that covers the domain Ω. The point-set $Y \subset \mathbb{R}^3$ is a set of sparse supports for the function $\bar{h}(\mathbf{x})$. This mesh size function incorporates both geometric and user-defined sizing constraints, and is *gradient-limited* to ensure that the Lipschitz conditions are satisfied. See Section 4.4 for further details regarding the construction of the mesh size functions used in this study.

A set of geometric structures, including the triangulated surface definitions Σ, mesh size functions $\bar{h}(\mathbf{x})$ and resulting volumetric meshes $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$ are shown in Figure 5.2 for a pair of example problems. In both cases, it can be seen that the mesh size

Figure 5.2: Geometrical structures for the BUNNY and VENUS surface meshing problems. The triangulated surfaces Σ are shown (left), contours of element size function $\bar{h}(\mathbf{x})$ are shown (centre) and surface meshes $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$ obtained using the Frontal-Delaunay algorithm are shown (right). Note that an interpolation of the size function onto the final mesh is shown, to facilitate the detailed cutaway views.



function $\bar{h}(\mathbf{x})$ reaches local minima at regions of high curvature or narrow separation in the underlying surface Σ. Additionally, the size function magnitude is seen to increase smoothly from these minima, reaching local maximum values in the 'interior' of the domains. The resolution of the corresponding volumetric meshes, in this case generated using the Frontal-Delaunay algorithm presented in Section 5.2, is clearly a good match to the contours of the associated mesh size functions, illustrating that the size-driven refinement schemes presented in previous sections is effective. A comparison of the final meshes $\mathcal{T}|_\Omega$ with the original triangulated surfaces Σ also demonstrates the success of the re-meshing paradigm used in this study, with a set of high-quality, graded volumetric meshes generated from low-quality initial surface triangulations of roughly constant resolution.

5.4 Domains with Sharp Features

The utility of both the Delaunay-refinement and Frontal-Delaunay algorithms presented in this chapter can be significantly improved by relaxing the constraints on the domain Ω, allowing the enclosing surface Σ to contain sharp ridges and creases. Clearly, for domains containing very sharp features, with dihedral angles $\theta \ll 90°$, it is not possible to ensure

that all elements in the resulting mesh $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$ satisfy the expected shape constraints, whilst concurrently enforcing domain conformity. Non-convergence may ensue if the unmodified Delaunay-refinement or Frontal-Delaunay algorithms are used to mesh domains containing sufficiently sharp features, with Steiner vertices positioned in the neighbourhood of such features potentially leading to an infinite cascade of mutual encroachment. When dealing with domains subtending small angles it is instead necessary to accept that some sacrifice of element quality is necessary in the vicinity of sharp geometric features and that special cases must be identified that allow a subset of low-quality elements adjacent to such features to persist in the final mesh $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$.

In the present study I adopt a strategy inspired by the so-called *protecting-balls* approach presented by Cheng, Dey and Ramos in [4]. Consistent with the detailed description of this technique presented in the context of surface mesh generation in Chapter 4, any sharp features in the domain Ω are enclosed in a set of Euclidean balls, within which refinement, through the introduction of new Steiner vertices, is prohibited. The protection strategy used within the volumetric meshing phase is inherited directly from the initial surface meshing pass, where any sharp peaks, ridges and creases in the domain Ω are identified and protected. The Delaunay-refinement and Frontal-Delaunay methods presented in Sections 5.1 and 5.2 are modified to test the suitability of any candidate Steiner vertices, declining to perform refinements that would result in the introduction of new vertices within a protecting ball. Such filtering is accomplished efficiently by storing the collection of protecting balls in a supporting AABB-tree. The protecting balls strategy clearly leads to the preservation of a subset of low-quality elements in the local neighbourhood of any sharp features in the Ω, and these elements appear in the final triangulation $\mathcal{T}|_\Omega = \text{Del}\,|_\Omega(X)$, in violation of the shape constraints $\rho(\tau) \geq \bar{\rho}$. I make no claims concerning the optimality of such an approach, but instead appeal to the experimental results presented in Section 5.6, demonstrating the effectiveness of such a strategy in practice.

5.5 Slivers, Optimality & Refinement Criteria

Contrary to the behaviour of the planar and surface meshing algorithms developed earlier in this thesis, the volumetric Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 5.1 and 5.2 do not ensure that bounds on the shape-quality of tetrahedral elements are satisfied. Such issues are known to manifest in practice, with volumetric meshes generated using conventional Delaunay-based techniques dogged by the occurrence of low-quality *sliver* elements with pathologically poor dihedral angles. This behaviour is well documented in the literature, including, for example, studies by Jamin et al. [12], Shewchuck [20, 21], Cheng, Dey, Edelsbrunner, Facello and Teng [3], Tournois, Srinivasanand and Alliez [23], and Si [22], amongst others. Cheng, Dey and Shewchuck provide an excellent overview of these issues in [5], summarising many of the individual contributions listed previously.

The lack of shape-optimality in higher dimensional tessellations stems from two important issues – a lack of topological optimality in the underlying Delaunay tessellation

itself, and the failure of the conventional radius-edge ratio to correctly characterise element geometry. Firstly, while it is known that higher dimensional Delaunay tessellations possess an array of interesting theoretical properties, as summarised in Chapter 2, the maximisation of element shape-quality is unfortunately not one of them. In $\mathbb{R}^2$, the Delaunay tessellation is known to minimise the radius of the largest circumdisk associated with the elements $\tau \in \text{Del}(X)$. Recalling that the circumradius, element geometry and plane angles are related via $\rho(\tau) = \frac{1}{2}(\sin(\theta_{\text{min}}))^{-1}$, it is clear that a minimisation of the circumradii leads to a maximisation of θ_{min}. In contrast, the situation in $\mathbb{R}^3$ is significantly less straightforward. Firstly, the Delaunay tessellation is known only to minimise the size of the maximum min-enclosing ball associated with elements $\tau \in \text{Del}(X)$, rather than the element circumballs. Furthermore, there is no relationship between the minimum element dihedral angles and the associated circumballs or min-enclosing balls. As a result, the distribution of angles for Delaunay tessellations in $\mathbb{R}^3$ is unbounded.

Remark 5.1 (optimality)**.** Higher-dimensional Delaunay tessellations $\text{Del}(X)$ for points $X \subset \mathbb{R}^d$ with $d \geq 3$ admit elements of pathologically low quality, even when X is *well-distributed*.

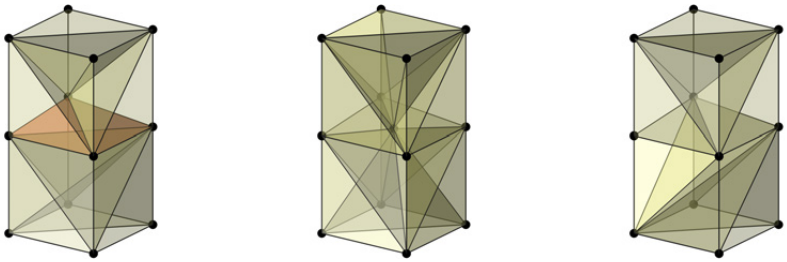
Unfortunately, it is known that a variety of *well-distributed* point-sets $X \subset \mathbb{R}^3$ support Delaunay tessellations incorporating elements of pathologically low shape-quality, with minimum and maximum dihedral angles approaching $0°$ and $180°$, respectively. The well-known sliver element is of particular concern, incorporating not only asymptotically poor dihedral angles, but also relatively small radius-edge ratios, making such elements undetectable to standard Delaunay-refinement strategies. In such cases, the four vertices of a tetrahedron τ are located close to the equatorial plane of the associated element circumball, resulting in a 'kite'-like geometry. Further details are provided in the catalogue of element configurations presented in Chapter 2.

In Figure 5.3, I present a simple example, designed to induce a Delaunay tessellation incorporating a sliver element. A array of twelve vertices is positioned in a lattice-like configuration, forming a structure equivalent to two 'cubes' stacked one upon the other, aligned with the $\mathbf{z}$-axis. Additionally, a single vertex located at the interface between the 'cubes' is perturbed in the $\hat{\mathbf{z}}$ direction by a small increment δ, ensuring that the interface between the two 'cubes' is a non-planar surface, marginally mis-aligned with the xy-plane. Importantly, note that $\delta \in \mathbb{R}$ is arbitrarily small. Analysis of the resulting Delaunay tessellation shows that a low-quality sliver element is generated adjacent to the perturbed vertex, and that the quality of this element is proportional to δ. Clearly as $\delta \to 0$ the minimum and maximum dihedral angles of the sliver approach $0°$ and $180°$, respectively. This example clearly illustrates the pitfalls of Delaunay-based mesh generation in $\mathbb{R}^3$, showing that a *well-separated* set of vertices can induce Delaunay tessellations incorporating elements of pathologically poor quality.

5.5.1 Alternative Refinement Criteria

In cases such as the example shown in Figure 5.3, a simple remedy is to re-visit the mechanisms used to drive shape-based refinement in Delaunay-based meshing algorithms.

Figure 5.3: A perturbed lattice configuration illustrating the occurrence of *sliver* elements. In (i) the standard Delaunay tessellation is shown, with a low-quality sliver element shown in red, wedged between the two adjacent 'cube'-like regions. In (ii) results of volume-length-driven refinement are shown, in which the sliver is eliminated through the introduction of a new Steiner vertex. Finally, in (iii), the topology of an optimal non-Delaunay tessellation is shown, free of low-quality tetrahedrons.



(i) (ii) (iii)

Conventionally, an element $\tau \in \text{Del}(X)$ is marked for refinement if its radius edge ratio exceeds a given threshold, such that $\rho(\tau) > \bar{\rho}$. Given that sliver elements are not detected and refined using such an approach, it is natural to seek out alternative strategies. Specifically, recent studies, including those presented by Gosselin and Olliver-Gooch in [11] and Cheng, Dey and Shewchuk in [5], investigate the effectiveness of refinement strategies driven by alternative definitions of element shape quality, including measures of the minimum element dihedral angles, volume-length ratios, aspect ratios and various other mixed metrics.

In the present study, I explore refinement schemes driven by element-wise volume-length ratios, in which elements with a volume-length measure $v(\tau)$ smaller than a prescribed threshold $\bar{v}$ are marked for refinement. Consistent with the various strategies presented in [11], the volume-length measure is known to be a *robust* metric in $\mathbb{R}^3$, detecting low-quality tetrahedra of all types. In contrast, quality metrics based solely on element angle distributions, such as those based on minimum dihedral angles [5], are known to admit certain classes of low-quality tetrahedra (*spear* types), and are therefore not sufficient when seeking to ensure that all low-quality elements are removed from a mesh. In [11], Gosselin and Olliver-Gooch present an extensive comparison of various alternative refinement schemes, and show that robust metrics, including the volume-length measure, offer the best overall performance.

In this study, I have implemented a volume-length driven strategy, modifying the Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 5.1 and 5.2 to refine any 3-simplexes $\tau \in \text{Del}\,|_\Omega(X)$ with small volume-length ratios $v(\tau) \leq \bar{v}$. Note that while these modified algorithms still prioritise element processing based on radius-edge ratios, the shape-based refinement strategy is driven purely by the element volume-length ratios. Such behaviour is achieved by setting $\bar{\rho} = \infty$. Note also, that the additional constraints, associated with element size, surface error and topology, are implemented as per the standard algorithms presented in Section 5.1 and 5.2. Application of this modified Delaunay-refinement algorithm to the 'lattice' example presented in Figure 5.3 leads to the insertion of an additional Steiner vertex about the mid-plane, eliminating

the adjacent low-quality sliver element.

Despite their apparent practical success, the development of robust theoretical guarantees remains an open problem for non-standard refinement schemes. Currently, convergence is not guaranteed, although, based on numerical experiments, Cheng, Dey and Shewchuk report in [5] that schemes based on the minimum element-wise dihedral angles reliably achieve dihedral angle distributions in the range $19° < \theta(\tau) < 153°$. In [11], Gosselin and Olliver-Gooch report similar bounds for schemes based on dihedral angle measures, in addition to bounds of $14° < \theta(\tau) < 154°$ for robust quality metrics. Results for the volume-length based scheme introduced in the present study are outlined in Section 5.6, showing good agreement with existing studies. Such results indicate that, while not proven, Delaunay-refinement schemes driven by non-standard quality metrics can be expected to converge in practice.

5.5.2 Topological Optimality & Non-Delaunay Tessellations

In contrast to the modified refinement schemes presented in the previous section, in which low quality tetrahedra are eliminated through additional refinement, considerations of *topological-optimality* offer an alternative pathway to improved element shape quality. Abandoning the sub-optimal Delaunay criterion, it is possible to search for alternative tessellations, designed to maximise a given element-wise cost function.

Remark 5.2 (alternative topology)**.** The Delaunay tessellation $\mathcal{T} = \text{Del}(X)$ of a point-set $X \subset \mathbb{X}^d$ with $d \geq 3$ is not guaranteed to maximise element shape-quality. An alternative, non-Delaunay topology $\mathcal{T}_{\text{ND}} \neq \mathcal{T}$, designed to maximise an element shape metric $\mathcal{Q}(\mathcal{T}_{\text{ND}})$, can lead to tessellations of improved shape-quality.

Consistent with the aim of maximising element shape-quality, cost functions based on element shape metrics, such as the minimum or maximum dihedral angles and/or volume-length measures, are obvious choices. The use of non-Delaunay tessellations, especially for problems in $\mathbb{R}^3$, is an idea that has been investigated in a number of previous studies. Specifically, a range of authors including Joe [13], Freitag and Ollivier-Gooch [9] and Klinger and Shewchuk [14, 15] have presented optimisation-based algorithms to generate pseudo-optimal tessellations for a given element-wise cost function. Additionally, the so-called *sliver-exudation* approach of Cheng et al. [3], makes use of the *weighted* Delaunay tessellation to improve the topology of the underlying mesh in the neighbourhood of low-quality sliver elements. In the present study, the notion of pseudo-optimal non-Delaunay tessellation is pursued in detail in Chapter 6, where these topological structures form the basis of a powerful mesh optimisation framework.

Returning to the 'lattice' example presented in Figure 5.3, an optimal non-Delaunay tessellation is presented, obtained by simply iterating over the set of candidate tessellations conforming to the given vertices and selecting the structure that resulted in a maximisation of the minimum element volume-length measure. Importantly, in contrast to the original Delaunay tessellation, note that the new tessellation is free of any low-quality elements. Furthermore, compared to the tessellation generated via the alternative volume-length based refinement scheme, the non-Delaunay tessellation achieves

high element shape-quality without the need for additional Steiner vertices. Such results demonstrate the potential of non-Delaunay-based mesh generation for higher-dimensional problems. While the development of non-Delaunay tetrahedral meshing algorithms is not pursued in the current study, these ideas are identified as an important avenue for future investigation.

5.6 Results & Discussion

The performance of the volumetric meshing algorithms introduced in this chapter was investigated experimentally, with both the Delaunay-refinement and Frontal-Delaunay algorithms developed in Sections 5.1 and 5.2 used to mesh a series of fifteen benchmark problems of varying size and complexity. Test domains were sourced from a range of application areas, including problems from computational modelling and simulation, computer graphics and medical imaging. Meshes for each domain, generated using the Frontal-Delaunay approach, are shown in Figure 5.4. Note that a number of test-cases include sharp surface ridges. Both the Frontal-Delaunay algorithm described in Section 5.2 and the Delaunay-refinement algorithm described in Section 5.1 have been implemented, allowing the performance and output of the two algorithms to be compared side-by-side. Due to the similarities in structure between the two algorithms, a common code-base is used, with the algorithms differing only in the type of Steiner vertices inserted and in the manner in which the queue of bad triangles is updated, as discussed in Section 5.2. Both algorithms are built using the `TRIPOD` and `LUMBERJACK` packages – making use of the efficient incremental Delaunay triangulation and spatial indexing frameworks presented in Chapter 2. Additionally, updates to the restricted surface and volumetric triangulations are achieved via an implementation of Algorithm 4.3.1, where the surface and volumetric intersection predicates are evaluated using double precision arithmetic. The Frontal-Delaunay and Delaunay-refinement algorithms are included in `JIGSAW` – a new mesh generation library built using the algorithms developed in this thesis. Both algorithms are implemented in C++ and compiled as 64-bit executables.

Meshes generated using the new Frontal-Delaunay algorithm for the full set of benchmarks are presented in Figure 5.4, demonstrating the effectiveness of the new strategy in practice. For all test problems, element quality has been catalogued, with normalised histograms of element volume-length $v(\tau)$, dihedral angle $\theta(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean volume-length measures, the worst-case angle bounds $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length. Meshes were generated using relatively coarse element size constraints, with the mesh size functions $\bar{h}(\mathbf{x})$ constructed using a Lipschitz smoothness parameter $g = 3/10$. Tight bounds on the element radius-edge ratios are imposed, such that $\bar{\rho} = 1.25$. While the shape of the surface facets in the embedded surface triangulation $\mathcal{T}|_\Sigma = \mathrm{Del}\,|_\Sigma(X)$ are bounded as a result, note that the dihedral angles of the 3-simplexes in the volumetric triangulation $\mathcal{T}|_\Omega = \mathrm{Del}\,|_\Omega(X)$ are unbounded – a major departure from the behaviour of the planar and surface meshing algorithms presented in previous chapters. An analysis of the results presented in Figure 5.4 shows, firstly, that the new Frontal-Delaunay al-

Figure 5.4: Benchmark problems for volumetric meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. A constant radius-edge ratio limit is specified, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

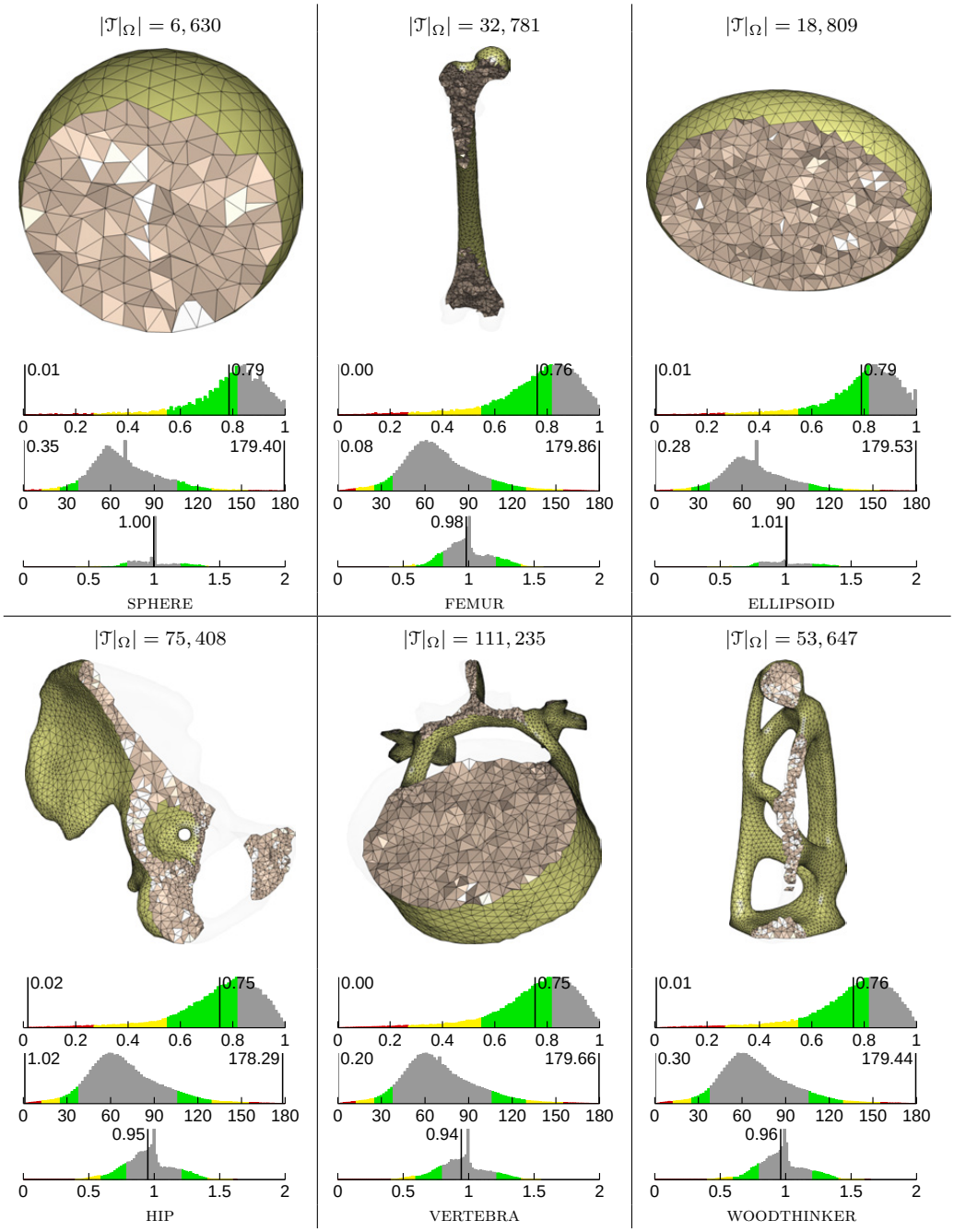


Figure 5.5: Benchmark problems for volumetric meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. A constant radius-edge ratio limit is specified, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

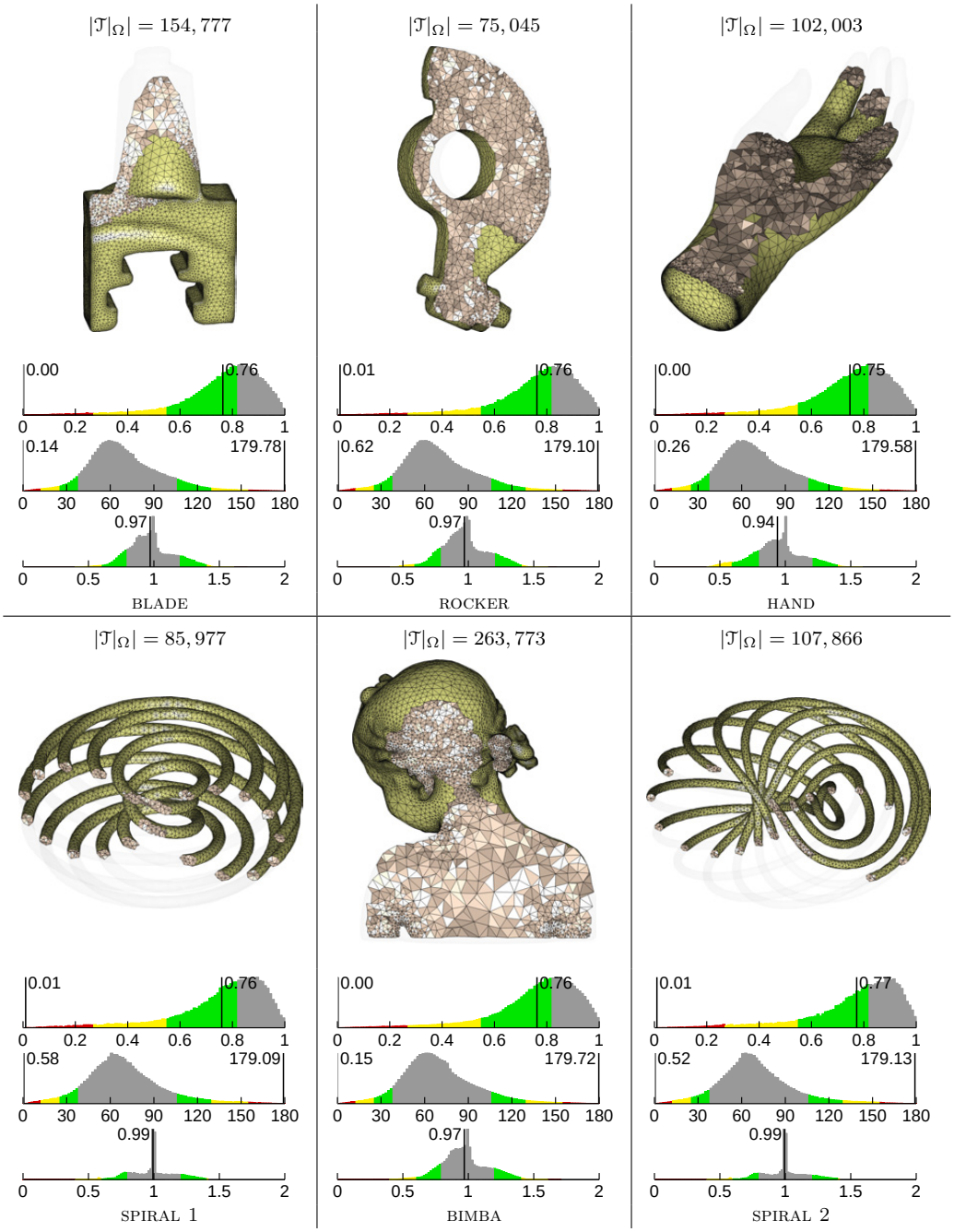
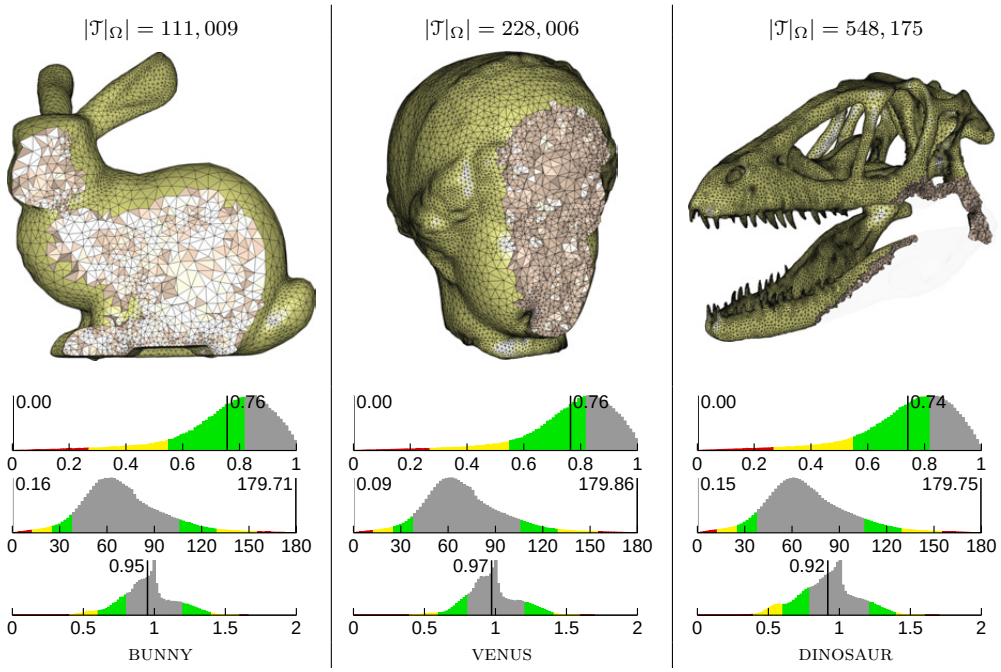


Figure 5.6: Benchmark problems for volumetric meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = {}^{3}/_{10}$. A constant radius-edge ratio limit is specified, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = ({}^{1}/_{10})\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.



gorithm is successful – meshing a range of complex volumetric domains, including those with sharp surface features. Furthermore, it is clear that the meshes respect the imposed size functions, with high resolution regions adjacent to regions of high curvature smoothly transitioning to sparser areas in the 'interior' of the domains. Secondly, it is noted that convergence is achieved for $\bar{\rho} \ll 2$, significantly outperforming the theoretical analysis presented in Section 5.1. Additionally, it is also important to note that convergence is achieved for a number of complex inputs containing sharp ridges in the enclosing surface Σ, with both the VENUS and DINOSAUR problems containing a number of these features with dihedral angles of under 30°. These results confirm the effectiveness of the *protecting-balls* strategy, used to support inputs with sharp features.

Inspection of the associated volume-length and dihedral angle distributions shown in Figure 5.4 shows that the lack of element shape bounds in $\mathbb{R}^3$ is not merely a theoretical concern, but an issue that significantly affects mesh quality in practice. It can be seen that a number of poorly shaped tetrahedrons are present in the meshes for all benchmark problems, and that a minority of these tetrahedrons are sliver elements, with pathologically poor dihedral angles approaching 0° and 180°. As per the discussions outlined in Section 1.1, the presence of these low-quality elements significantly limits the applicability of these meshes for use in subsequent numerical modelling and simulation studies.

5.6.1 Comparative Performance

The results of a comparative performance study, contrasting the effectiveness of the Frontal-Delaunay and Delaunay-refinement meshing algorithms, is presented in Figures 5.7 and 5.8. Detailed results for the BUNNY and VENUS test problems are examined, investigating the performance of both algorithms for a range of different input parameters. Specifically, the effectiveness of the new size-driven point-placement scheme is addressed, comparing a range of meshes generated by both the Frontal-Delaunay and Delaunay-refinement algorithms in terms of their size- and shape-quality and underlying structure. In addition to the detailed results presented for the BUNNY and VENUS problems, a simplified set of comparisons are also presented for the full set of benchmark problems.

5.6.1.1 Size-driven Refinement

A detailed study of meshes generated for the BUNNY and VENUS problems, presented in Figures 5.7 and 5.8, examines the impact of the size-optimal Type II off-centres introduced in Section 5.2. A set of meshes were generated for test cases using low, medium and high resolution settings, where the associated mesh size functions $\bar{h}(\mathbf{x})$ were constructed using Lipschitz smoothness values of $g = 3/10$, $g = 2/10$ and $g = 1/10$, respectively. The radius-edge ratio limit was held constant across all test cases, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold was enforced for the embedded surface triangulation $\mathcal{T}|_\Sigma = \mathrm{Del}\,|_\Sigma(X)$, setting $\bar{\epsilon} = \beta\bar{h}(\mathbf{x})$, with $\beta = 1/10$. For all test problems, element quality has been catalogued, with normalised histograms of element volume-length $v(\tau)$, dihedral angle $\theta(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean volume-length measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length. Note that the domain for the VENUS test case includes a number of sharp ridges in the underlying surface Σ, located near the neck of the statue.

Analysis of Figures 5.7 and 5.8 shows that both the Frontal-Delaunay and Delaunay-refinement algorithms successfully generate meshes in all cases, but, as per discussions in the preceding sections, the lack of element shape optimality is obvious, with a number of low-quality sliver elements generated in each case. Inspection of distributions of $v(\tau)$, $\theta(\tau)$ and $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ show that the Frontal-Delaunay algorithm consistently outperforms the Delaunay-refinement scheme, generating meshes with higher mean volume-length ratios and 'tight' distributions of relative edge length in all cases. The most significant difference in behaviour between the two algorithms appears to be in the way that mesh size constraints are imposed. Given the narrow distributions of relative edge length, strongly clustered about $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$, it is clear that the Frontal-Delaunay algorithm successfully generates meshes accurately conforming to the imposed mesh size constraints. In contrast, output generated using the Delaunay-refinement scheme is seen to incorporate significant sizing error, typified by 'broad' distributions of relative edge length straddling $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)} = 1$. While the mean relative edge lengths are comparable in all cases, the Frontal-Delaunay algorithm clearly generates much more accurate output on an element-

Figure 5.7: Size-driven refinement study for the BUNNY problem, showing meshes generated using the Frontal-Delaunay (upper) and Delaunay-Refinement (lower) algorithms. Meshes are generated with increasing values of size function smoothness, $g = 3/10$, $g = 2/10$ and $g = 1/10$ from left to right. A constant radius-edge ratio limit is specified for all cases, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, such that $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

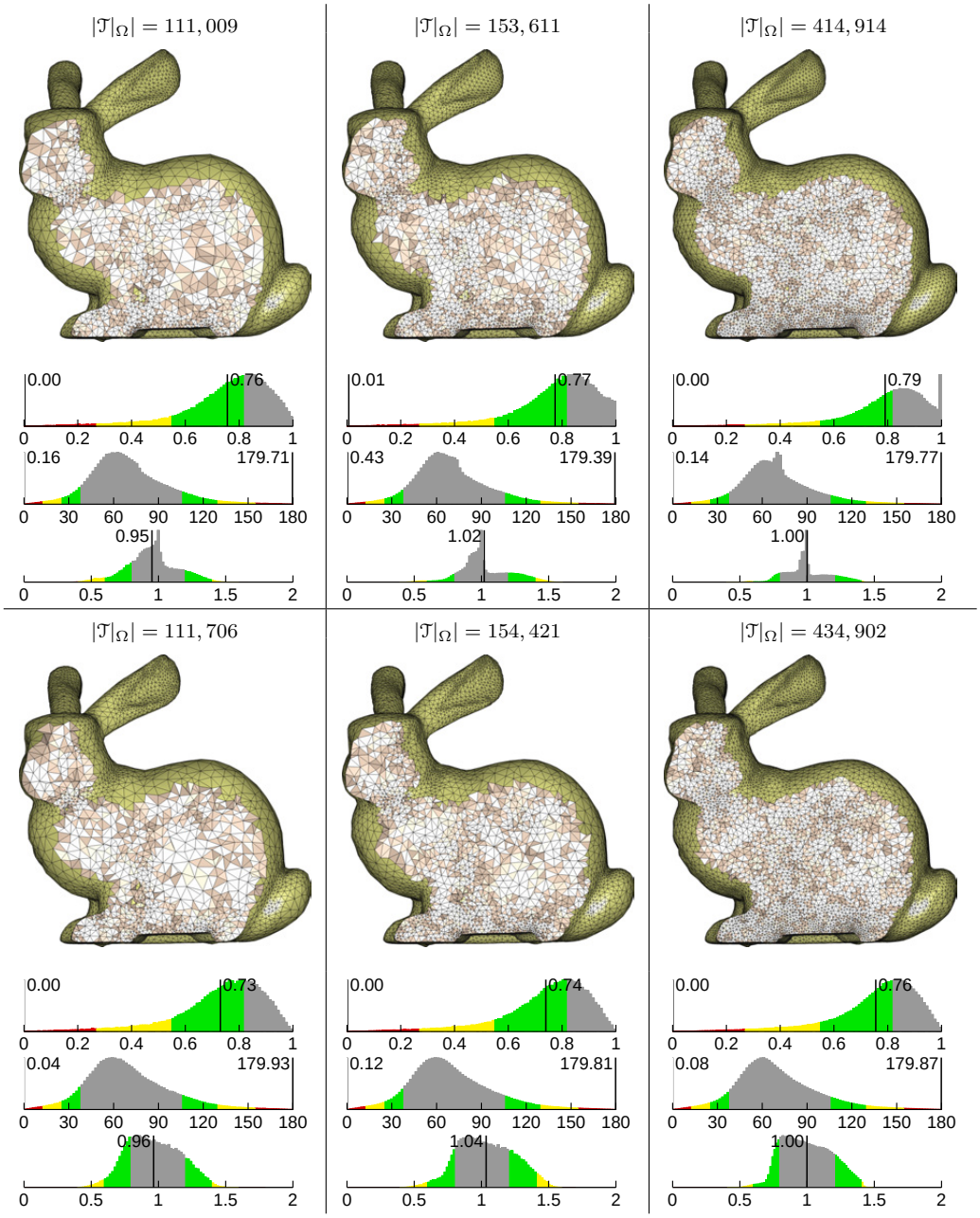


Figure 5.8: Size-driven refinement study for the VENUS problem, showing meshes generated using the Frontal-Delaunay (upper) and Delaunay-Refinement (lower) algorithms. Meshes are generated with increasing values of size function smoothness, $g = 3/10$, $g = 2/10$ and $g = 1/10$ from left to right. A constant radius-edge ratio limit is specified for all cases, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, such that $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

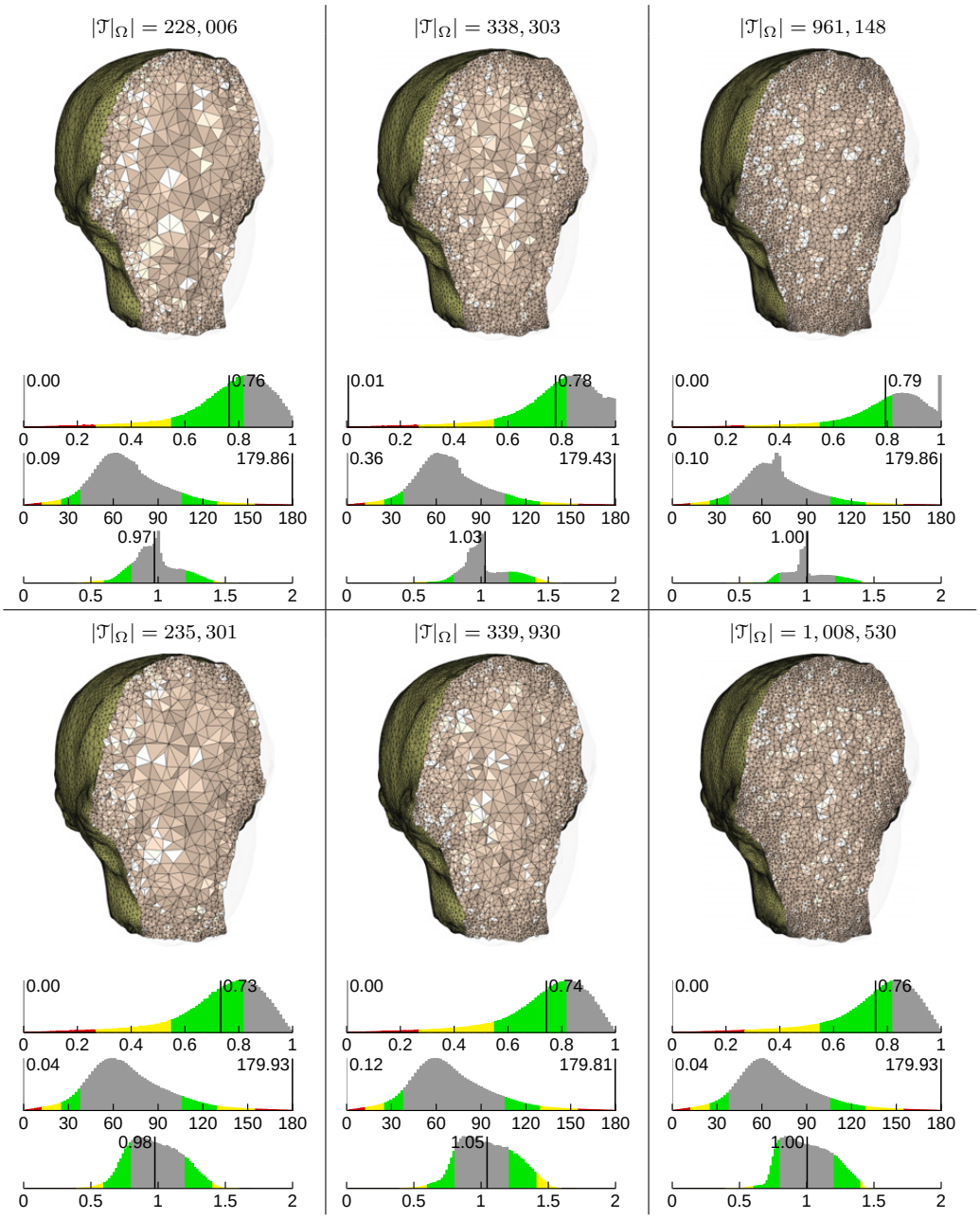


Table 5.1: Volume meshing study, comparing the performance of the Frontal-Delaunay and Delaunay-refinement algorithms on the full set of benchmark problems. Meshes are generated using medium resolution settings, such that the mesh size functions $\bar{h}(\mathbf{x})$ are constructed with $g = 2/10$. A constant radius-edge ratio limit is specified for all cases, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, such that $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Results listed include: total element count $|\mathcal{T}|_\Omega|$, total runtime $t(s)$, mean and minimum volume-length ratios $\overline{v(\tau)}$, $v(\tau)_{\min}$ and dihedral angle bounds $\theta(\tau)_{\min}$, $\theta(\tau)_{\max}$.

Domain	Frontal-Delaunay						Delaunay-refinement					
	$\lvert\mathcal{T}\rvert_\Sigma\rvert$	$t(s)$	$v(\tau)_{\min}$	$\overline{v(\tau)}$	$\theta(\tau)_{\min}$	$\theta(\tau)_{\max}$	$\lvert\mathcal{T}\rvert_\Sigma\rvert$	$t(s)$	$v(\tau)_{\min}$	$\overline{v(\tau)}$	$\theta(\tau)_{\min}$	$\theta(\tau)_{\max}$
SPHERE	5,668	0.19	0.01	0.79	0.62°	178.9°	5,900	0.18	0.07	0.76	3.86°	174.1°
ELLIPSOID	16,337	0.48	0.01	0.79	0.38°	179.3°	16,971	0.45	0.01	0.75	0.87°	178.5°
FEMUR	41,878	2.26	0.00	0.77	1.11°	177.8°	42,805	2.12	0.00	0.74	0.30°	179.5°
HIP	105,607	3.66	0.00	0.77	0.09°	179.8°	109,859	3.43	0.00	0.73	0.05°	179.9°
VERTEBRA	135,097	4.33	0.00	0.76	0.05°	179.9°	137,746	3.98	0.00	0.73	0.05°	179.9°
WOOD	63,939	2.91	0.00	0.77	0.90°	178.4°	67,342	2.75	0.01	0.74	0.29°	179.5°
HAND	156,581	7.67	0.00	0.77	0.08°	179.9°	161,099	7.28	0.00	0.74	0.15°	179.7°
BLADE	210,409	8.83	0.00	0.77	0.26°	179.6°	216,008	8.36	0.00	0.74	0.11°	179.8°
ROCKER	93,138	3.41	0.00	0.77	0.08°	179.9°	95,764	3.16	0.00	0.74	0.06°	179.9°
SPIRAL 1	76,676	3.29	0.00	0.77	0.70°	178.8°	80,041	3.05	0.00	0.73	0.15°	179.7°
SPIRAL 2	96,798	3.97	0.00	0.79	0.30°	179.5°	100,026	3.72	0.00	0.73	0.07°	179.9°
BUNNY	153,611	5.36	0.00	0.77	0.50°	179.4°	154,421	4.92	0.00	0.74	0.04°	179.9°
BIMBA	384,533	19.27	0.00	0.78	0.10°	179.8°	390,664	17.88	0.00	0.74	0.13°	179.8°
VENUS	338,303	16.86	0.00	0.78	0.40°	179.4°	339,930	15.81	0.00	0.74	0.05°	179.9°
DINOSAUR	690,574	34.57	0.00	0.76	0.20°	179.7°	688,504	29.91	0.00	0.73	0.04°	179.9°

by-element basis.

Additionally, analysis of the distributions of $v(\tau)$ and $\theta(\tau)$ shows that, despite the presence of these low-quality elements, the Frontal-Delaunay algorithm also consistently outperforms the Delaunay-refinement scheme in terms of shape quality, generating meshes with higher mean volume-length ratios in all cases. Furthermore, it is evident that the quality of meshes generated using the Frontal-Delaunay algorithm improves as $g \to 0$, as indicated by the increase in mean $v(\tau)$ and the narrowing of the $\theta(\tau)$ distribution about $70.5°$[1]. Specifically, it can be seen that sharp peaks about $v(\tau) \simeq 1$ and $\theta(\tau) \simeq 70.5°$ manifest as $g \to 0$, showing that a significant percentage of near-perfect elements are generated by the Frontal-Delaunay algorithm as the mesh size constraints become more restrictive. Similar analysis shows that the Delaunay-refinement results are only weakly dependent on the Lipschitz smoothness of the mesh size function, with mean $v(\tau)$ increasing only marginally. No detectable narrowing of the distribution of $\theta(\tau)$ is evident, in contrast to the behaviour of the Frontal-Delaunay algorithm. Visually, the enhanced quality of the meshes generated using the Frontal-Delaunay algorithm is evident, with marked increases in mesh smoothness and sub-structure obvious. Meshes generated by both algorithms are seen to be similar size, with neither algorithm showing a significant or consistent deviation in element count $|\mathcal{T}|_\Omega|$.

Analysis of the Steiner refinement strategies logged throughout this study show that the majority of the off-centres chosen by the Frontal-Delaunay algorithm are either size-optimal Type II points ($\simeq 55\%$) or conventional Type I circumcentres ($\simeq 45\%$), exhibiting similar ratios to those observed for the planar algorithm presented in Chapter 3. The bias toward size-optimal off-centres is also observed to increase as $\bar{h}(\mathbf{x}) \to 0$. These results are not surprising, simply indicating that when the magnitude of the mesh size function $\bar{h}(\mathbf{x})$ is sufficiently small, the element size constraints dominate.

5.6.1.2 Overall Comparisons

In addition to the detailed comparisons presented previously, a simplified set of comparative results were also obtained for all benchmark problems, contrasting the performance of the Frontal-Delaunay and Delaunay-refinement approaches. The results of this study are presented in Table 5.1. Meshes were generated using medium resolution settings, with the mesh size function $\bar{h}(\mathbf{x})$ constructed with $g = 2/10$. Again, tight bounds on element radius-edge ratios were specified, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold was enforced, setting $\bar{\epsilon} = \beta\bar{h}(\mathbf{x})$, with $\beta = 1/10$. A consistent mesh size function was used when generating both the embedded surface and volumetric triangulations. Analysis of the results presented in Table 5.1 confirm the trends observed in the detailed analysis carried out for the BUNNY and VENUS problems – that, given an appropriate mesh size function $\bar{h}(\mathbf{x})$, the proposed Frontal-Delaunay algorithm produces meshes that are of higher shape- and size-quality than those generated using the conventional Delaunay-refinement algorithm. These results are confirmed by the increase in mean volume-length ratio v_{mean} for meshes generated using the Frontal-Delaunay

[1] A regular tetrahedron has a uniform dihedral angle of $\simeq 70.53°$

algorithm. Consistent with the results presented in previous sections, analysis of the worst-case dihedral angles confirms that all meshes generated using both the Delaunay-refinement and Frontal-Delaunay methods include low quality sliver elements. Note that the extreme angles are typically equally poor for both methods.

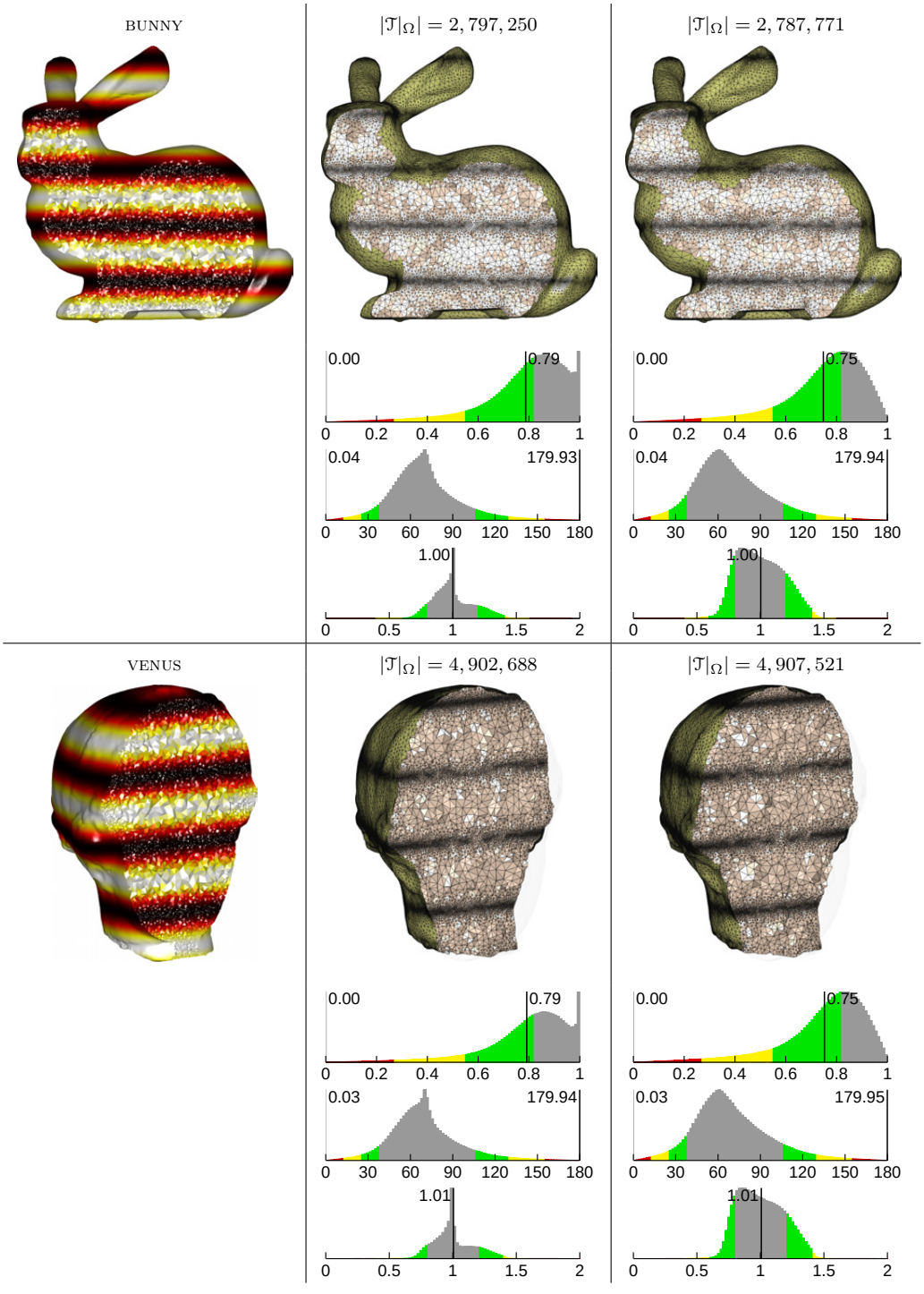
Total run-times for the algorithms are also tabulated and show, firstly, that the implementations developed in this study are efficient, generating meshes containing $1,000,000$'s of elements in a matter of minutes. Consistent with the behaviour of the surface algorithms presented in Chapter 4, it can be seen that the surface Frontal-Delaunay algorithm is typically marginally slower than the Delaunay-refinement method. This additional computational burden is associated with the use of off-centres in the Frontal-Delaunay algorithm, specifically, the placement of size-optimal Type II vertices according to the iterative strategy described in Sections 4.2 and Sections 5.2. Such a strategy requires the solution of a small local system of non-linear equations, which, in the case of surface refinement, leads to the iterative computation of surface intersections. Such intersections are relatively costly, currently implemented as geometric searches over the facets of the triangulated surface Σ via traversals of a supporting AABB-tree. The development of techniques to speed-up this process is flagged as an avenue for future research.

5.6.2 User-defined Size Constraints

The performance of the volumetric meshing algorithms for problems involving user specified mesh size functions $\bar{h}_u(\mathbf{x})$ was also assessed. In Figure 5.9, the results of a graded meshing study are presented, in which a set of volumetric meshes were generated for the BUNNY and VENUS test problems that adhere to user defined sizing constraints. The user defined constraints in these examples are illustrative-only, consisting of a simple sinusoidal variation, $\bar{h}_u(\mathbf{x}) = c_1 \sin(c_2 \mathbf{z}) + c_3$, where the coefficients $c_1, c_2, c_3 \in \mathbb{R}$ are chosen based on the dimensions of the geometry and $\mathbf{z}$ is a locally aligned vertical coordinate. Note that in addition to the user specified size function $\bar{h}_u(\mathbf{x})$, a geometric size function $\bar{h}_g(\mathbf{x})$ was also constructed, with the final mesh size function $\bar{h}(\mathbf{x})$ taken as the minimum of the user-defined and geometric contributions, such that $\bar{h}(\mathbf{x}) = \min\big(\bar{h}_g(\mathbf{x}), \bar{h}_u(\mathbf{x})\big)$, as per the methods outlined in Section 4.4. A consistent mesh size function was used when generating both the embedded surface and volumetric triangulations. The meshes generated using both the Frontal-Delaunay and Delaunay-refinement algorithms are presented in Figure 5.9. The final mesh size function $\bar{h}(\mathbf{x})$ was constructed using a Lipschitz smoothness parameter $g = 2/10$. In all cases, a constant radius-edge ratio was specified, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold was enforced for the embedded surface triangulation $\mathcal{T}|_\Sigma = \text{Del}\,|_\Sigma(X)$, setting $\bar{\epsilon} = \beta \bar{h}(\mathbf{x})$, with $\beta = 1/10$. For all test problems, element quality has been catalogued, with normalised histograms of element volume-length $v(\tau)$, dihedral angle $\theta(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean volume-length measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length.

Analysis of these graded meshes confirm much of the behaviour discussed in the previous comparisons, with the Frontal-Delaunay algorithm generating meshes of higher

Figure 5.9: Adaptive meshing study for the BUNNY and VENUS test cases, showing (left) contours of the user-defined mesh size functions $\bar{h}_u(\mathbf{x})$. Meshes generated using the Frontal-Delaunay (centre) and Delaunay-refinement (right) algorithms are graded to conform to these size constraints. A constant radius-edge ratio limit is specified for all cases, such that $\bar{\rho} = 1.25$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Sigma|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.



mean shape- ans size-quality when compared to the conventional Delaunay-refinement method. Analysis of the distributions of $v(\tau)$ and $\theta(\tau)$ reveals that sharp peaks, corresponding to $v(\tau) \simeq 1$ and $\theta(\tau) \simeq 70.5°$, are present in results for the Frontal-Delaunay algorithm, confirming that a large fraction of near-perfect elements are generated. Corresponding results for the Delaunay-refinement scheme, on the other hand, do not exhibit such behaviour. Careful analysis of Figure 5.9 also shows that meshes generated using the Frontal-Delaunay technique typically conform more uniformly to the specified mesh size function, resulting in smoother meshes with more regular substructure. Lastly, consistent with preceding volumetric results, the lack of tetrahedral element shape optimality is clearly visible, with a minority of low-quality sliver elements present in all meshes. Overall, despite the sliver elements, these results demonstrate the effectiveness of the proposed Frontal-Delaunay method when generating graded volumetric meshes for user-defined sizing constraints.

5.6.3 Alternative Refinement Criteria

In an effort to improve worst-case element shape-quality, the alternative volume-length-based refinement strategy presented in Section 5.5 has been implemented. As discussed previously, this new refinement scheme is a simple variation on the standard radius-edge-driven techniques used in both the Delaunay-refinement and Frontal-Delaunay algorithms analysed in the preceding sections. In the new scheme, rather than refining elements based on their radius-edge ratios, all elements $\tau \in \mathrm{Del}\,|_\Omega(X)$ are selected for refinement if their volume-length ratios are too small, such that $v(\tau) \leq \bar{v}$, where $\bar{v}$ is a user-defined constant $\bar{v} \in [\,0, 1\,]$. The implementation of the new refinement strategy requires only minimal modification to the existing Delaunay-refinement and Frontal-Delaunay algorithms – simply requiring that an additional comparison of volume-length ratios be performed when assessing element refinement status. While the existing radius-edge ratio infrastructure is maintained, and is used to prioritise the refinement schedule, explicit radius-edge-driven refinement is disabled by setting $\bar{\rho} = \infty$. The modified algorithms are otherwise consistent with the Delaunay-refinement and Frontal-Delaunay algorithms presented in Sections 5.1 and 5.2, inheriting an identical treatment of the additional element size, surface error and topological constraints.

In Figure 5.10, meshes generated using the modified Frontal-Delaunay algorithm are presented, clearly demonstrating the effectiveness of the new volume-length refinement strategy in eliminating low-quality elements. In addition to cutaway views of the resulting tetrahedral structures, element quality is also catalogued, with normalised histograms of element volume-length $v(\tau)$, dihedral angle $\theta(\tau)$ and relative edge length $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ illustrated. Histograms further highlight the minimum and mean volume-length measures, the worst-case angle bounds, $\theta_{\min}$ and $\theta_{\max}$ and the mean relative edge length. Results are presented for the full set of benchmark problems previously processed using the 'standard' Frontal-Delaunay algorithm in Figure 5.4, allowing direct side-by-side comparisons between the new volume-length and existing radius-edge-driven refinement schemes to be conducted. Furthermore, meshes were generated using consistent input parameters,

Figure 5.10: Benchmark problems for volumetric meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. Contrary to previous results, refinement is driven using a threshold on element volume-length ratios only, such that $\bar{v} = 1/3$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

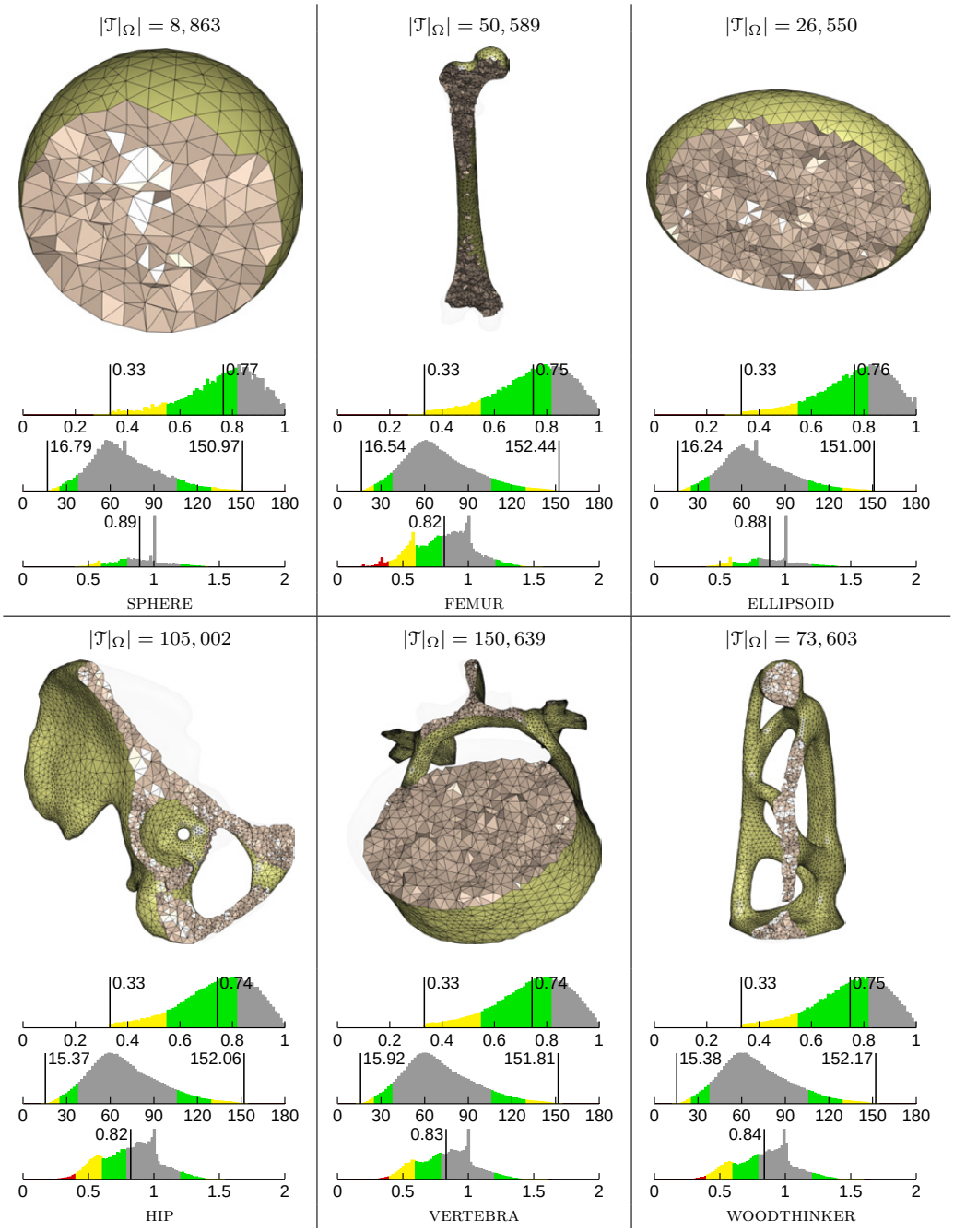


Figure 5.11: Benchmark problems for volumetric meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. Contrary to previous results, refinement is driven using a threshold on element volume-length ratios only, such that $\bar{v} = 1/3$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.

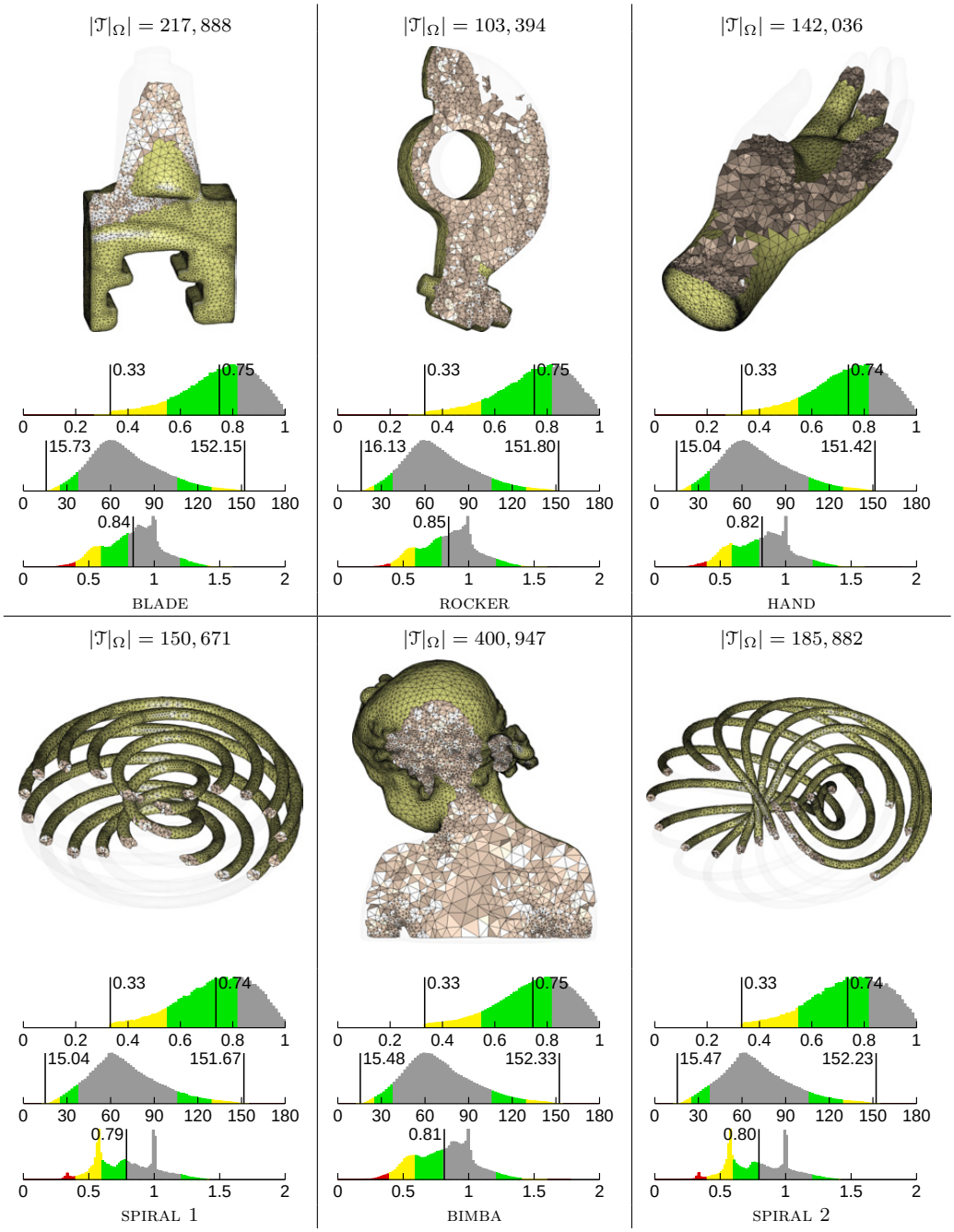
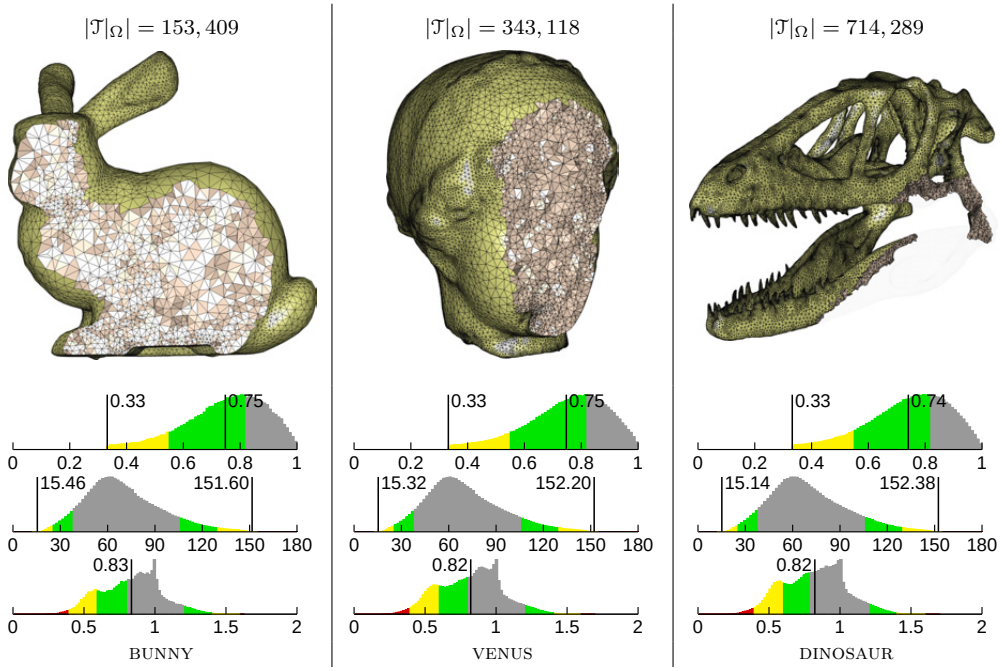


Figure 5.12: Benchmark problems for volumetric meshing studies, showing meshes generated using the Frontal-Delaunay algorithm with coarse settings, such that $g = 3/10$. Contrary to previous results, refinement is driven using a threshold on element volume-length ratios only, such that $\bar{v} = 1/3$. A non-uniform surface discretisation threshold is imposed, with $\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$. Element counts $|\mathcal{T}|_\Omega|$ are included for each case. Normalised histograms of element volume-length ratio $v(\tau)$, dihedral angle $\theta(\tau)$ and relative size $\frac{\|\mathbf{e}\|}{\bar{h}(\mathbf{x}_e)}$ are also shown.



with identical sizing constraints ($g = 3/10$) and surface error thresholds ($\bar{\epsilon} = (1/10)\bar{h}(\mathbf{x})$) imposed. In this study, I have found that convergence was achieved for all test problems when using a volume-length threshold $\bar{v} \leq 1/3$. For larger threshold values as small as $\bar{v} = 0.34$, non-convergence was observed across the board, with failures reported for all types of problem, even simple geometries such as the SPHERE and ELLIPSOID test cases. I believe these results indicate that $\bar{v} = 1/3$ is a reliable upper-bound on the allowable volume-length threshold, applicable to volumetric meshing problems in general. In addition to these experimental results, the development of a theoretical model of convergence, and the associated derivation of bounds for $\bar{v}$ is clearly of considerable practical interest and may be a fruitful topic for future investigations.

An analysis of the results presented in Figure 5.10 shows, firstly, that the new volume-length refinement strategy is successful in practice – achieving uniform convergence for a range of complex volumetric domains, including those with sharp surface features. Furthermore, in contrast with meshes generated using radius-edge-based refinement shown in Figure 5.4, it is clear that the new strategy significantly improves the shape quality of the worst-case elements in the resulting meshes, capping the element volume-length measures such that $v(\tau) > 1/3$. Corresponding improvements in the dihedral angle distributions are also observed, with the worst-case angles lying in the range $14^\circ < \theta(\tau) < 153^\circ$ for all test problems. These results support those presented by Gosselin and Ollivier-Gooch in [11]. In [5], Cheng, Dey and Shewchuk report dihedral angle distributions as good

as $19^\circ < \theta(\tau) < 153^\circ$ for large-scale meshing problems, though it is important to note that their alternative refinement scheme is based on the dihedral angles only, admitting the occurrence of other types of low-quality tetrahedra with poor *plane* angles, as catalogued in Section 5.5. Note also that, compared to measures of the dihedral angles, the volume-length measure is computationally inexpensive to evaluate.

Further analysis of Figures 5.4 and 5.10 reveals that the improved shape quality achieved using the new volume-length-driven refinement strategy comes at the cost of increased mesh size. Comparisons of the output sizes $|\mathcal{T}|_\Omega = \mathrm{Del}\,|_\Omega(X)|$ shows that larger meshes are generated in all cases when using the new volume-length based scheme. For some test problems, such as the SPIRAL3 case, the increase in size is significant, with the mesh generated using the volume-length refinement strategy approximately 1.6 times larger than the equivalent results generated using the standard algorithm. On average, the increase is size is typically more subdued, with most meshes 1.2–1.3 times larger when refined using the new volume-length scheme. Analysis of the distributions of relative edge length $\frac{\|\mathbf{e}\|}{h(\mathbf{x}_e)}$ confirms that the volume-length refinement strategy leads to non-uniform grading, in which elements in the output mesh $\mathcal{T}|_\Omega$ are smaller than the local mesh size constraints. Such behaviour is consistent with results presented by Cheng, Dey and Shewchuk in [5].

While volume-length driven refinement is clearly effective in producing bounded-quality tessellations in practice, the increased mesh size and sub-optimal mesh grading are considered to be non-negligible drawbacks. In spite of this, such methods clearly improve the utility of the volumetric meshing algorithms developed in this chapter, facilitating the construction of meshes that satisfy a range of size, shape and approximation related constraints and guarantees. In contrast to meshes built using standard Delaunay-refinement techniques, such meshes are directly applicable to problems in computational simulation and modelling. Alternative schemes, based on geometrical and topological optimisation strategies, are presented in the next chapter. Consistent with the findings of Gosselin and Ollivier-Gooch [11], such methods are considered to comprise a better solution to the problem of quality tetrahedral mesh generation.

5.7 Conclusions

In this chapter, I have presented a pair of algorithms designed to generate high-quality meshes for volumetric domains enclosed by 2-manifold surfaces. Both algorithms are based on the so-called *restricted* Delaunay triangulation, in which an embedded surface triangulation $\mathcal{T}|_\Sigma = \mathrm{Del}\,|_\Sigma(X)$ and corresponding volumetric triangulation $\mathcal{T}|_\Omega = \mathrm{Del}\,|_\Omega(X)$, both conforming to an underlying surface Σ, are constructed as subsets of a full-dimensional Delaunay tessellation $\mathrm{Del}(X)$. The first algorithm is a 'conventional' restricted Delaunay-refinement scheme, closely modelled on the `CGALMESH` algorithm presented by Jamin et al. in [12], in which an embedded surface triangulation is first constructed following the Delaunay-refinement methods outlined in Chapter 4. The subsequent volumetric triangulation is built using a variation of Shewchuk's scheme [20, 21], whereby the circumcentres of poor quality tetrahedrons are inserted into the mesh as

Steiner vertices. In the second algorithm, I have proposed a new restricted Frontal-Delaunay technique, generalising the ideas introduced in Chapters 3 and 4 for planar and surface meshing algorithms to support volumetric domains. In this new algorithm, I again initially generate an embedded surface triangulation, using the Frontal-Delaunay algorithm outlined in Chapter 4. The subsequent volumetric triangulation is built using generalised off-centre Steiner vertices, in which elements are refined by carefully inserting new points along segments in the associated Voronoi diagram. Off-centre Steiner vertices are positioned based on a combination of size- and shape-driven refinement strategies, leading to a hybrid approach that combines many of the advantages of advancing-front and Delaunay-refinement techniques. A series of comparative experimental studies confirm the effectiveness of this new approach, demonstrating that an improvement in element quality is typically achieved when comparing the new Frontal-Delaunay method with conventional Delaunay-refinement techniques. Importantly, it has also been demonstrated that the new Frontal-Delaunay algorithm achieves the same theoretical optimality as the conventional Delaunay-refinement approach, satisfying constraints on element radius-edge ratios, edge length and surface discretisation thresholds. Despite these theoretical guarantees, the lack of element shape optimality for tetrahedral elements is identified as a major weakness for both the Delaunay-refinement and Frontal-Delaunay algorithms, with meshes generated by these algorithms in practice seen to contain a minority of poor-quality sliver elements. The genesis of these elements is traced to two underlying issues: (i) the lack of topological optimality for the Delaunay tessellation in dimensions higher than 2, and (ii) the failure of the element-wise radius-edge ratio to detect low-quality sliver tetrahedrons.

In addition to algorithms based on standard radius-edge refinement strategies, I have also developed an alternative refinement criteria, designed to detect and eliminate all classes of low-quality tetrahedrons. This new method uses the element volume-length measure, rather than the conventional radius-edge ratio, to drive shape-based refinement, converging when all elements in a mesh satisfy $v(\tau) > \bar{v}$, where $\bar{v}$ is a user-defined constant. Based on a comprehensive set of experimental studies, an upper bound $\bar{v} \leq 1/3$ has been established for which convergence can be expected in practice. The development of an associated theoretical model of convergence remains an open problem. A series of experimental studies confirm the effectiveness of this new approach in practice, demonstrating that modified Delaunay-refinement and Frontal-Delaunay algorithms, incorporating the new volume-length-based refinement strategy, successfully refine a series of complex volumetric benchmark problems without the introduction of low-quality tetrahedral elements. The cost of this new refinement technique is an increase in mesh size, with experimental studies confirming that output size is increased by a factor of approximately 1.2 on average and 1.6 in the worst-case for the test problems examined in this study. Despite the increase in mesh size, algorithms based on the new volume-length refinement strategy represent a significant improvement over conventional methods, facilitating the generation of high-quality volumetric meshes suitable for computational modelling and simulation.

References

[1] Boissonnat, J. D., and Oudot, S. Provably Good Surface Sampling and Approximation. In *ACM International Conference Proceeding Series* (2003), vol. 43, pp. 9–18.

[2] Boissonnat, J. D., and Oudot, S. Provably Good Sampling and Meshing of Surfaces. *Graphical Models 67*, 5 (2005), 405–451.

[3] Cheng, S. W., Dey, T. K., Edelsbrunner, H., Facello, M. A., and Teng, S. H. Silver Exudation. *J. ACM 47*, 5 (Sept. 2000), 883–904.

[4] Cheng, S. W., Dey, T. K., and Ramos, E. A. Delaunay Refinement for Piecewise Smooth Complexes. *Discrete & Computational Geometry 43*, 1 (2010), 121–166.

[5] Cheng, S. W., Dey, T. K., and Shewchuk, J. R. *Delauay Mesh Generation.* Taylor & Francis, New York, 2013.

[6] Chernikov, A. N., and Chrisochoides, N. P. Generalized insertion region guides for Delaunay mesh refinement. *SIAM Journal on Scientific Computing 34*, 3 (2012), A1333–A1350.

[7] Engwirda, D. Size-optimal Steiner points for Delaunay-refinement in volumetric domains. *Submitted to Procedia Engineering: 24th International Meshing Roundtable* (2015).

[8] Foteinos, P. A., Chernikov, A. N., and Chrisochoides, N. P. Fully generalized two-dimensional constrained Delaunay mesh refinement. *SIAM Journal on Scientific Computing 32*, 5 (2010), 2659–2686.

[9] Freitag, L. A., and Ollivier-Gooch, C. Tetrahedral Mesh Improvement using Swapping and Smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[10] Frey, P. J., Borouchaki, H., and George, P. L. 3D Delaunay Mesh Generation Coupled with an Advancing-Front Approach. *Computer Methods in Applied Mechanics and Engineering 157*, 12 (1998), 115 – 131.

[11] Gosselin, S., and Ollivier-Gooch, C. Tetrahedral Mesh Generation using Delaunay Refinement with Non-standard Quality Measures. *International Journal for Numerical Methods in Engineering 87*, 8 (2011), 795–820.

[12] Jamin, C., Alliez, P., Yvinec, M., and Boissonnat, J. D. CGALmesh: A Generic Framework for Delaunay Mesh Generation. Tech. rep., INRIA, 2013.

[13] Joe, B. Construction of Three-dimensional Improved-Quality Triangulations Using Local Transformations. *SIAM Journal on Scientific Computing 16*, 6 (1995), 1292–1307.

[14] Klingner, B. M. *Tetrahedral Mesh Improvement.* PhD thesis, Berkeley, California, 2008.

[15] Klingner, B. M., and Shewchuk, J. R. Aggressive Tetrahedral Mesh Improvement. In *Proceedings of the 16th International Meshing Roundtable*, M. L. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 3–23.

[16] Mavriplis, D. J. An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *Journal of Computational Physics 117*, 1 (1995), 90 – 101.

[17] Mavriplis, D. J. Unstructured Grid Techniques. *Annual Review of Fluid Mechanics 29*, 1 (1997), 473–514.

[18] Oudot, S., Rineau, L., and Yvinec, M. Meshing Volumes Bounded by Smooth Surfaces. In *Proceedings of the 14th International Meshing Roundtable* (2005), Springer, pp. 203–219.

[19] Rebay, S. Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal of Computational Physics 106*, 1 (1993), 125 – 138.

[20] Shewchuk, J. R. *Delaunay Refinement Mesh Generation*. PhD thesis, Pittsburg, Pennsylvania, 1997.

[21] Shewchuk, J. R. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry* (New York, NY, USA, 1998), SCG '98, ACM, pp. 86–95.

[22] Si, H. Adaptive Tetrahedral Mesh Generation by Constrained Delaunay Refinement. *International Journal for Numerical Methods in Engineering 75*, 7 (2008), 856–880.

[23] Tournois, J., Srinivasan, R., and Alliez, P. Perturbing Slivers in 3D Delaunay Meshes. In *Proceedings of the 18th International Meshing Roundtable*, B. Clark, Ed. Springer Berlin Heidelberg, 2009, pp. 157–173.

Chapter 6

Mesh Improvement

In this chapter, I develop a new framework for mesh improvement that is designed to enhance the quality of existing simplicial meshes. I focus on the development of a general algorithm that is equally applicable to each of the the planar, surface and volumetric mesh types investigated in this thesis. Mesh improvement is fundamentally an optimisation-based task, seeking to maximise a given objective function through the application of geometrical and topological updates. Achieving globally optimal mesh configurations is known to be difficult, requiring the solution of a coupled geometrical and topological optimisation problem. In this work, I focus instead on the development of a local optimisation strategy, in which a 'good' locally-optimal solution is sought based on a given initial mesh configuration. The mesh improvement process is driven by an element-wise *mesh-quality* metric – a scalar function that scores the shape-quality of each element based on its geometry.

Extending previous work due primarily to Freitag and Ollivier-Gooch [13] and Klinger and Shewchuk [19, 20], I build a mesh improvement framework utilising a variety of local optimisation *predicates*, including: (i) local vertex smoothing operations, based on a perturbation of vertex coordinates, (ii) topological transformations, via local updates to the underlying mesh connectivity, and (iii) vertex insertion, in which additional Steiner vertices are introduced to the mesh. Consistent with previous approaches, I show that a locally-optimal solution can be achieved through an iterative application of these optimisation predicates, with convergence attained when no further improvement in mesh quality is possible. I investigate the effectiveness of a new priority-driven optimisation schedule, in which optimisation effort is focused on regions of the mesh that are amenable to further improvement. As such, I pursue an 'active-set' philosophy, in which, within each iteration, the optimisation predicates are applied to a restricted subset of the mesh entities that are identified as candidates for further optimisation. The evolution of this active-set is tracked explicitly as the optimisation proceeds. I show that such an approach can lead to significant improvements in computational efficiency while preserving optimisation performance.

The effectiveness of the new mesh improvement framework is assessed experimentally, examining its effectiveness when applied to a comprehensive set of benchmark problems,

including a range of planar, surface and volumetric test cases. Additionally, a series of comparative studies are undertaken, contrasting the performance of the new tetrahedral mesh improvement scheme with the existing mesh optimisation package `STELLAR`, developed by Klinger and Shewchuk [19, 20]. Experiments are conducted using a range of complex benchmark problems, verifying the robustness, efficiency and practical performance of the proposed schemes.

6.1 Mesh Improvement Strategies

Irrespective of the manner in which a computational mesh is generated, the *quality* of the underlying tessellation can often be enhanced, sometimes significantly, through the application of so-called *mesh improvement* strategies. Such high-quality meshes are desirable in that they facilitate the construction of optimal discretisations for the numerical solution of partial differential equations, as per the discussions presented in Chapter 1. Due to difficulties associated with the direct generation of high-quality tetrahedral meshes, caused in part by the presence of low-quality sliver elements, mesh improvement is known to be especially important for volumetric problems.

Mesh improvement is typically viewed as an optimisation problem, requiring that a geometrical and topological configuration be found that optimises a given mesh *quality function* $\mathcal{Q}(\tau)$, designed to score the configuration of each element $\tau \in \mathcal{T}$ based on a set of user-defined criteria. In this thesis, I restrict my interest to a straightforward metric, based solely on the shape quality of each element. The resulting mesh improvement process can therefore be expected to optimise element shape quality only. Note that such simplifications are unnecessary in the general case, with a number of authors, including, for example, Freitag and Knupp [12], Klinger [19] and Frey and Alauzet [14] previously investigating composite mesh quality functions designed to optimise a range of characteristics, including the element size, shape, orientation and anisotropy. In this work, I further restrict the choice of quality metric to the so-called area- and volume-length measures presented in Chapter 2. Such formulations are known to robustly detect all types of low-quality triangular and tetrahedral element types, in contrast to other metrics, such as the Delaunay criterion, or measures based on the dihedral angles. Recent studies, such as the work of Klinger and Shewchuk [19, 20] have shown that use of the volume-length measure often leads to optimal mesh improvement results for a wide range of tetrahedral test cases.

6.2 Mesh Improvement Predicates

Consistent with the approaches advocated by Freitag and Ollivier-Gooch [13] and Klinger and Shewchuk [20], the mesh improvement algorithms developed in this thesis are built upon a set of mesh improvement *predicates* – a collection of core mesh transformation operations designed to improve the shape-quality of local element 'patches' within a given mesh. These predicates include a range of topological transformations, geometrical operations and refinement-based templates. Following closely the work of Klinger and

Shewchuk [20], I implement all optimisation predicates in a so-called *hill-climbing* fashion, accepting the geometrical or topological modifications dictated by a particular operation only if the local mesh quality metrics are sufficiently improved. Specifically, I adopt a *worst-first* strategy, in which a given optimisation predicate is required to improve the worst-case quality metric within the local element subset being acted upon. Such a philosophy ensures that the mesh quality metric is increased monotonically as the optimisation proceeds. This behaviour aims to maximise the minimum quality metric in the mesh, rather than improving a mean measure.

Before moving on to a detailed discussion of the various optimisation predicates used in this thesis, I introduce a number of constructs used to define the *local neighbourhood* for a given entity in a tessellation. The use of such neighbourhoods is integral to the development of the disparate topological, geometrical and insertion-based predicates developed in subsequent sections.

Definition 6.1 (topological distance)**.** Given an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the *topological distance* $\mathcal{D}(v_i, v_j)$ between two vertices $v_i, v_j \in \mathcal{V}$ is the minimum number of edges in a path between v_i and v_j.

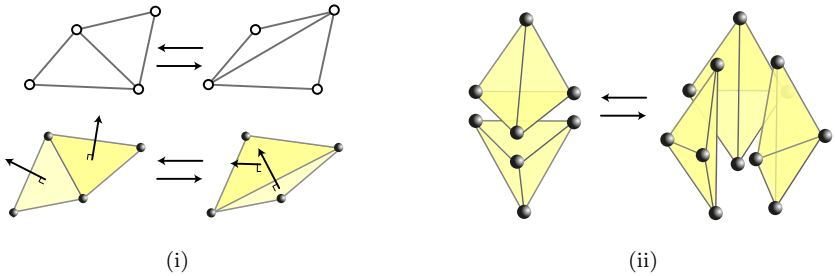
Definition 6.2 (k-ring neighbourhood)**.** Given an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, the *k-ring neighbourhood* of a vertex $v_i \in \mathcal{V}$ is the set of vertices $\{v_1, v_2, \ldots, v_n\} \subseteq \mathcal{V}$ with a topological distance $\mathcal{D}(v_k, v_i)$ less than or equal to k.

Recalling that the topology of a simplicial tessellation can be represented as an undirected graph, it is natural to consider local entity neighbourhoods via the k-ring construction. In this work, the 1-ring neighbourhood is frequently used, to identify, for example, the subset of elements at unit topological distance from a given vertex, edge or element. In this case, the neighbourhoods simply consist of those entities immediately adjacent to the given entity. General k-ring neighbourhoods can be found by performing a suitable breadth-first-search in the underlying adjacency graph.

6.2.1 Topological Transformations

The first class of mesh improvement predicates explored in this work concerns the topological optimality of the underlying tessellation itself. Following a strategy originally presented by Joe [16, 17], I construct a *catalogue* of local topological transformations, that are designed to incrementally improve the underlying topology of a given mesh. Such transformations, or *flips*, are local in nature, spanning only a small, densely connected subset of the overall mesh topology. Given their local nature, a sequence of such transformations are typically required, iterating over the collection of possible element subsets in a given mesh. The topological transformations presented in this work can be thought of as local *re-triangulation* operations, in which the elements associated with a local convex *cavity* $\mathcal{C}$ embedded in a mesh $\mathcal{T}$ are removed and replaced with a new conforming tessellation of $\mathcal{C}$ that improves the distribution of local quality metrics. It is important to note that, given an initial Delaunay tessellation, $\mathcal{T} = \mathrm{Del}(X)$, such operations do not necessarily preserve the Delaunay criterion, but instead seek to form a

Figure 6.1: Illustration of bistellar flips for triangular and tetrahedral tessellations, showing: (i) the $\mathcal{T}_2\mathcal{T}_2$ edge-flip for planar (top) and surface (bottom) trinagulations, and (ii) the $\mathcal{T}_2\mathcal{T}_3$ and $\mathcal{T}_3\mathcal{T}_2$ face- and edge-flips for tetrahedral complexes.



(i) (ii)

locally optimal tessellation. In the case of tetrahedral meshes, where it is known that the Delaunay criterion leads to the formation of sub-optimal sliver elements, such behaviour is often desirable.

6.2.1.1 Bistellar Flips

Bistellar flips are the simplest type of topological transformation that can be applied to simplicial meshes. For triangulations embedded in $\mathbb{R}^2$, such operations are commonly known as *edge-flips*, whereby the common edge adjacent to two neighbouring triangular elements is 'flipped', connecting the opposing vertices in the associated quadrilateral cavity. Such an operation is referred to as a $\mathcal{T}_2\mathcal{T}_2$ flip in this thesis, indicating the number of associated elements before and after the flipping operation is performed. Consistent with the *hill-climbing* nature of the mesh improvement framework, the $\mathcal{T}_2\mathcal{T}_2$ flip seeks to improve the worst-case element quality metric, such that $\min(\mathcal{Q}(\mathcal{T}_i'), \mathcal{Q}(\mathcal{T}_j')) > \min(\mathcal{Q}(\mathcal{T}_i), \mathcal{Q}(\mathcal{T}_j))$, where $\mathcal{T}_i', \mathcal{T}_j'$ denote the 'flipped' element configuration. The edge flip operation in $\mathbb{R}^2$ is illustrated in detail in Figure 6.1.

The edge flip operation is also applicable to 2-manifold surface triangulations embedded in $\mathbb{R}^3$, in which the non-planar quadrilateral cavity associated with adjacent 2-simplexes is re-triangulated by flipping the common diagonal edge. In such a configuration, in addition to considerations of local metrics, care is needed to ensure that the transformed mesh remains homeomorphic to the original surface topology, requiring that both the orientation and manifold-ness of the mesh is preserved. In the current work, the 2-manifold property is preserved by testing the degree of the flipped edge, ensuring that the new edge is shared between a maximum of two elements in the triangulation. An additional orientation test is also performed in the flipped configuration, ensuring that the angle θ_z, formed between adjacent element normals $\mathbf{z}_i', \mathbf{z}_j'$, is sufficiently small, such that $\theta_z \leq \beta$, where β is a user-defined threshold. In this thesis, the orientation angle threshold is constrained to the range $\beta \in [0, 90°]$. The embedded edge flip operation in $\mathbb{R}^3$ is illustrated in detail in Figure 6.1.

Bistellar flips for tetrahedral elements in $\mathbb{R}^3$ are significantly more complicated than their lower-dimensional counterparts, consisting of a pair of inverse edge-face operations. Originally documented by Lawson [21] and further analysed by Joe [16], such

operations are constructed by considering the number of possible triangulations of 5 vertices in $\mathbb{R}^3$. Lawson showed that such configurations support two possible triangulations, consisting of either two or three tetrahedrons. As a result, a pair of transformations are induced, known as the $\mathcal{T}_2\mathcal{T}_3$ and $\mathcal{T}_3\mathcal{T}_2$ operations in this thesis. Given a pair of tetrahedra sharing a common 2-face f, *sandwiched* between a pair of opposing vertices a, b, the $\mathcal{T}_2\mathcal{T}_3$ operation attempts to create three new tetrahedra that share a common edge ab. Such an operation seeks to improve the worst-case quality metric, such that $\min(\mathcal{Q}(\mathcal{T}'_i), \mathcal{Q}(\mathcal{T}'_j), \mathcal{Q}(\mathcal{T}'_k)) > \min(\mathcal{Q}(\mathcal{T}_i), \mathcal{Q}(\mathcal{T}_j))$, where $\mathcal{T}'_i, \mathcal{T}'_j, \mathcal{T}'_k$ denote the 'flipped' element configuration. Conversely, given an interconnected triplet of tetrahedra sharing a common edge ab, the $\mathcal{T}_3\mathcal{T}_2$ operation attempts to create a pair of tetrahedra adjacent to the vertices a, b that share a common 2-face f. It is clear that such an operation is an inverse for the $\mathcal{T}_2\mathcal{T}_3$ flip described previously. Again, such an operation is designed to improve the worst-case quality metric , such that $\min(\mathcal{Q}(\mathcal{T}'_i), \mathcal{Q}(\mathcal{T}'_j)) > \min(\mathcal{Q}(\mathcal{T}_i), \mathcal{Q}(\mathcal{T}_j), \mathcal{Q}(\mathcal{T}_k))$, where $\mathcal{T}'_i, \mathcal{T}'_j$ denote the 'flipped' element configuration. The $\mathcal{T}_2\mathcal{T}_3$ and $\mathcal{T}_3\mathcal{T}_2$ bistellar flips are illustrated in detail in Figure 6.1.

6.2.1.2 Edge Removal

The *edge-removal* operation, first introduced by Briére de L'isle and George [2] is a topological transformation for tetrahedral meshes in which a single common edge, shared by a *ring* of adjacent tetrahedra $\mathcal{M} \subseteq \mathcal{T}$, is removed. The resulting cavity is re-triangulated about the non-planar polygon $\mathcal{R}$, formed by the set of opposing edges in the sub-mesh $\mathcal{M}$. The edge-removal operation can be thought of as a generalisation of the simple $\mathcal{T}_3\mathcal{T}_2$ flip, and, as a result, is known as the $\mathcal{G}_3\mathcal{G}_2$ operation in this work. The edge-removal operation has previously been used to great effect in the context of tetrahedral mesh improvement by a number of authors, including de Cougny and Shephard [6], Freitag, Ollivier-Gooch [13] and Klinger and Shewchuk [20].

Given an existing edge ab, shared by a ring of tetrahedra $\mathcal{M}$, the $\mathcal{G}_3\mathcal{G}_2$ transformation seeks to build a new triangulation $\mathcal{N}$ that improves the minimum local element quality metric, such that $\min(\mathcal{Q}(\mathcal{N})) > \min(\mathcal{Q}(\mathcal{M}))$. The new tetrahedral sub-mesh $\mathcal{N}$ is constructed by selecting an appropriate triangulation $\mathcal{T}_\mathcal{R}$ of the non-planar polygonal ring $\mathcal{R}$, allowing the 'upper' and 'lower' halves of the cavity to be re-triangulated as $\mathcal{N} = \bigcup_{t_i \in \mathcal{T}_\mathcal{R}} \{\text{Conv}(t_i, \{a\}), \text{Conv}(t_i, \{b\})\}$. The edge-removal process is illustrated in detail in Figure 6.2. For $|\mathcal{M}| > 3$, the triangulation $\mathcal{T}_\mathcal{R}$ is clearly non-unique, making selection of an appropriate $\mathcal{T}_\mathcal{R}$ non-trivial. Specifically, it is required that a particular $\mathcal{T}_\mathcal{R}$ be selected that leads to a maximisation of the minimum quality metric associated with the new set of tetrahedra $\mathcal{N}$.

Selecting an optimal $\mathcal{T}_\mathcal{R}$ efficiently is the key challenge when implementing the $\mathcal{G}_3\mathcal{G}_2$ operation in practice. Following the methods detailed by Freitag and Ollivier-Gooch [13], I employ a 'lookup-table' approach in this work, in which the complete combinatorial spaces associated with triangulations of the ring $\mathcal{R}$ are stored in a set of static tables. The optimal $\mathcal{T}_\mathcal{R}$ is then selected by simply iterating over elements in the candidate triangulations, keeping track of the minimum quality metric in each case. The size of these static tables increases exponentially with increasing $|\mathcal{M}|$, and, for considerations

of efficiency, I restrict the $\mathcal{G}_3\mathcal{G}_2$ operation to relatively low-degree configurations in this thesis, for which $|\mathcal{M}| \leq 8$. Note that, in practical cases, such a constraint is not expected to be overly restrictive, with both Freitag and Ollivier-Gooch [13] and Shewchuk [23] noting that $\mathcal{G}_3\mathcal{G}_2$ operations are rarely successful in practice for configurations in which $|\mathcal{M}| > 8$. Static combinatorial tables representing the candidate triangulations $\mathcal{T}_\mathcal{R}$ can be formed via the Hurtardo-Noy hierarchy [15], a tree-like representation enumerating the set of possible triangulations associated with a given convex polygonal region. Note that, due to the size of the combinatorial space, implementations of the Hurtardo-Noy hierarchy exhibit inherently exponential complexity.

Restrictions to low-degree configurations can be partially ameliorated through an approach of Shewchuk [23], who selects the optimal ring triangulation $\mathcal{T}_\mathcal{R}$ using a polynomial-time, dynamic programming algorithm of Klincsek [18]. While the $O\big(|\mathcal{M}|^3\big)$ performance of such an approach is asymptotically superior to the simple lookup table methodology used in the current work, such differences are not expected to manifest significantly for the small $|\mathcal{M}| \leq 8$ typically encountered in practice.

6.2.1.3 Multi-Face Removal

The *multi-face-removal* operation, introduced by de Cougny and Shephard [6], is a topological transformation for tetrahedral meshes in which a *ring* of 2-faces *sandwiched* between a pair of vertices are replaced by a single edge. The resulting cavity $\mathcal{C}$ is re-triangulated about the new edge. The multi-face-removal operation can be thought of as the inverse of the edge-removal operation $\mathcal{G}_3\mathcal{G}_2$, and, as a result is known as the $\mathcal{G}_2\mathcal{G}_3$ operation in this work. Such a transformation can also, equivalently, be thought of as a generalisation of the simple $\mathcal{T}_2\mathcal{T}_3$ flip. The multi-face-removal operation seems to be somewhat under-utilised, appearing only in the original unpublished report of de Cougny and Shephard, and the recent studies of Klinger and Shewchuk [20] and Misztal, Bærentzen, Anton and Erleben [22].

Given a vertex pair a, b, the $\mathcal{G}_2\mathcal{G}_3$ operation proceeds by identifying a set of 2-faces $\mathcal{T}_\mathcal{R} \subseteq \mathcal{T}$ that are *sandwiched* between the vertices a, b, such that each 2-face $t_i \in \mathcal{T}_\mathcal{R}$ is shared by a pair of tetrahedra adjacent to the vertices a, b. Given such a configuration, a $\mathcal{G}_2\mathcal{G}_3$ operation attempts to replace the set of tetrahedra $\mathcal{N}$ adjacent to a subset of the sandwiched faces $f \subseteq \mathcal{T}_\mathcal{R}$ by re-triangulating the ring-shaped cavity associated with f about the edge ab, such that $\min(\mathcal{Q}(\mathcal{M})) > \min(\mathcal{Q}(\mathcal{N}))$, where $\mathcal{M}$ is the set of new tetrahedra. This re-triangulation step is an exact inverse for the edge-removal operation described in previous sections. The multi-face-removal procedure is shown in detail in Figure 6.2.

Given a set of sandwiched faces $\mathcal{T}_\mathcal{R}$, an optimal $\mathcal{G}_2\mathcal{G}_3$ operation selects the subset $f \subseteq \mathcal{T}_\mathcal{R}$ that leads to a maximisation of the minimum quality metric for the set of new tetrahedra $\mathcal{M}$. Shewchuk [23] has shown that such a selection can be performed in $O(|\mathcal{T}_\mathcal{R}|)$ time via a tree-based traversal of the local adjacency graph of $\mathcal{T}_\mathcal{R}$, leading to an optimal $O(|\mathcal{N}|)$ time $\mathcal{G}_2\mathcal{G}_3$ algorithm overall. Shewchuk's traversal algorithm is implemented in the current work without modification. Interestingly, note that while an efficient linear-time algorithm is available for the multi-face-removal operation, the

Figure 6.2: Illustration of the edge- and multi-face removal operations for tetrahedral complexes, showing the sequence of inverse operations required to transform from one state to another. Note that both the $\mathcal{G}_2\mathcal{G}_3$ and $\mathcal{G}_3\mathcal{G}_2$ predicates act about the non-planar polygonal *ring* $\mathcal{T}_\mathcal{R}$. In the case of the $\mathcal{G}_3\mathcal{G}_2$ edge-removal, the existing edge ab is replaced with an optimal triangulation $\mathcal{T}_\mathcal{R}$. The inverse $\mathcal{G}_2\mathcal{G}_3$ multi-face removal replaces the existing triangulation $\mathcal{T}_\mathcal{R}$ by splitting about the new edge ab.

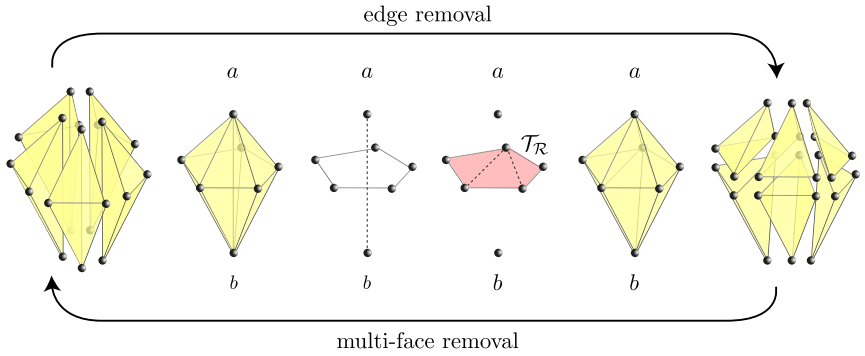
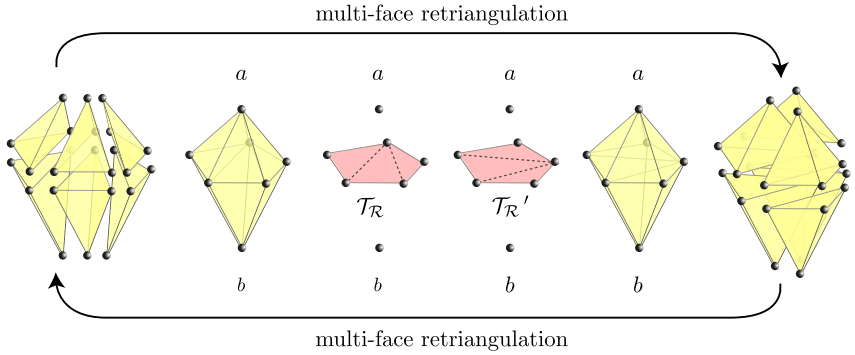


Figure 6.3: Illustration of the multi-face re-triangulation operations for tetrahedral complexes, showing the sequence of inverse operations required to transform from one state to another. Like the edge- and multi-face removal operations, multi-face re-triangulation acts about the non-planar polygonal *ring* $\mathcal{T}_\mathcal{R}$, seeking to find an improved triangulation $\mathcal{T}_\mathcal{R}'$. Note that contrary to the edge- and multi-face removal operations, multi-face re-triangulation is its own inverse.



existence of such an approach for the inverse edge-removal transformation is unknown.

6.2.1.4 Multi-Face Retriangulation

The *multi-face retriangulation* operation, introduced by Misztal, Bærentzen, Anton and Erleben [22], is a topological transformation for tetrahedral meshes in which the tetrahedra adjacent to a *ring* of 2-faces *sandwiched* between a pair of vertices is re-triangulated, *preserving* the sandwiched region. Such an operation is related to the $\mathcal{G}_2\mathcal{G}_3$ and $\mathcal{G}_3\mathcal{G}_2$ flips outlined previously, and is denoted as the $\mathcal{G}_4\mathcal{G}_4$ operation in this work. The multi-face retriangulation operation is, to my knowledge, investigated only in the recent study of Misztal et al. [22], where it was shown to lead to moderate improvements in mesh quality compared to comparative results generated using a combinations of edge-removal and multi-face-removal operations alone.

Given a vertex pair a, b, the $\mathcal{G}_4\mathcal{G}_4$ operation proceeds by identifying a set of 2-faces $\mathcal{T}_\mathcal{R} \subseteq \mathcal{T}$ that are *sandwiched* between the vertices a, b, such that each 2-face $t_i \in \mathcal{T}_\mathcal{R}$

is shared by a pair of tetrahedra adjacent to both a, b. This process is equivalent to that described previously for the $\mathcal{G}_2\mathcal{G}_3$ operation. Given such a configuration, the $\mathcal{G}_4\mathcal{G}_4$ operation attempts to replace the set of tetrahedra $\mathcal{N}$ adjacent to the sandwiched faces $\mathcal{T}_\mathcal{R}$ by finding a new optimal triangulation $\mathcal{T}'_\mathcal{R}$ of the ring-shaped region $\mathcal{R}$, such that $\min(\mathcal{Q}(\mathcal{M})) > \min(\mathcal{Q}(\mathcal{N}))$, where $\mathcal{M}$ is the set of new tetrahedra adjacent to $\mathcal{T}'_\mathcal{R}$. Like the $\mathcal{G}_3\mathcal{G}_2$ operation, such a process seeks to retriangulate the 'upper' and 'lower' halves of the cavity as $\mathcal{M} = \bigcup_{t_i \in \mathcal{T}_\mathcal{R}} \{\text{Conv}(t_i, \{a\}), \text{Conv}(t_i, \{b\})\}$. The multi-face retriangulation process is illustrated in detail in Figure 6.3.

In contrast to the $\mathcal{G}_2\mathcal{G}_3$ and $\mathcal{G}_3\mathcal{G}_2$ operations discussed previously, which seek to replace a set of sandwiched faces with an edge and vice-versa, the $\mathcal{G}_4\mathcal{G}_4$ operation simply seeks to retriangulate an existing set of sandwiched faces. Consistent with the $\mathcal{G}_3\mathcal{G}_2$ operation outlined in previous sections, such a procedure involves the selection of an optimal triangulation $\mathcal{T}_\mathcal{R}$ that leads to a maximisation of the minimum quality metric associated with the new set of tetrahedra $\mathcal{M}$. In this work, such an operation is implemented for $\mathcal{N} \leq 8$ via the same static lookup table approach used for the $\mathcal{G}_3\mathcal{G}_2$ operation described previously. As per the earlier discussions, Shewchuk [23] has shown that such a process can be implemented in $O\big(|\mathcal{N}|^3\big)$ time using a dynamic programming approach of Klincsek [18].

6.2.2 Locally Optimal Tessellation

Given a catalogue of topological transformations, the notion of locally optimal tessellations can be explored. Following an approach of Joe [16, 17], I define such an object to be any tessellation that is locally optimal with respect to its available catalogue of transformations.

Definition 6.3 (locally optimal tessellation)**.** Let $\mathcal{T}$ be a tessellation of a point-set $X \subset \mathbb{R}^d$ and let Θ be a *catalogue* of topological transformations. Let $\mathcal{Q}$ be an element-wise objective function. The tessellation $\mathcal{T}$ is said to be *locally-optimal* if no *quality-improving* transformation $\theta_i \in \Theta$ can be found for the tessellation $\mathcal{T}$. A transformation θ_i is said to be quality-improving if $\mathcal{Q}(\theta_i(\mathcal{T})) > \mathcal{Q}(\mathcal{T})$ where $\theta_i(\mathcal{T})$ is the updated topology induced by the transformation θ_i.

In the absence of global criteria, such as the Delaunay property, the notion of local optimality is a convenient framework within which the quality of a general tessellation can be assessed. Such tessellations can be constructed by exhaustive application of the set of topological transformations, achieving convergence when no further transformations can be successfully applied. In the context of mesh optimisation, topological optimality is often pursued via relatively naive approaches, with a number of authors, including Joe [17], Freitag and Ollivier-Gooch [13] and Klinger and Shewchuk [20] simply iteratively applying global sweeps of transformations until convergence is obtained. I pursue an alternative queue-based strategy that is expected to significantly reduce the computational effort required to find locally optimal tessellations.

Given an initial tessellation $\mathcal{T}$ and a catalogue of Θ of topological operations, a locally optimal tessellation is sought via a greedy, queue-based procedure. Such an algorithm

Algorithm 6.2.1 Local Topology Optimisation

```
1: function OptimTopology(X, T)                          ▷ {find locally optimal tessellation}
2:     Initialise queue of active elements P ← T.
3:     while (P ≠ ∅) do
4:         if (TestFlip(t_i ← P)) then                   ▷ {element-specific flips}
5:             Update queue P ← θ_i(t_i).
6:         end if
7:     end while
8: end function
```

```
1: function TestFlipSimplex2(t_i)                        ▷ {flips for 2-simplex elements}
2:     for all (e_j ∈ t_i) do                            ▷ {edge-based flips}
3:         if (TestFlipT2T2(t_i, e_j)) then
4:             Update topology T ← T_2T_2(T)
5:             return True.
6:         end if
7:     end for
8:     return False.
9: end function
```

```
1: function TestFlipSimplex3(t_i)                        ▷ {flips for 3-simplex elements}
2:     for all (f_j ∈ t_i) do                            ▷ {face-based flips}
3:         Find opposing vertex pair a, b for face f_j.
4:         if (TestFlipG2G3(t_i, a, b)) then
5:             Update topology T ← G_2G_3(T)
6:             return True.
7:         end if
8:         if (TestFlipG4G4(t_i, a, b)) then
9:             Update topology T ← G_4G_4(T)
10:            return True.
11:        end if
12:    end for
13:    for all (e_j ∈ t_i) do                            ▷ {edge-based flips}
14:        if (TestFlipG3G2(t_i, e_j)) then
15:            Update topology T ← G_3G_2(T)
16:            return True.
17:        end if
18:    end for
19:    return False.
20: end function
```

maintains a priority queue $\mathcal{P}$ of *active* elements, about which the full catalogue Θ of topological transformations are attempted. At each iteration, the active element $t_i \in \mathcal{P}$ at the front of the queue is removed and the sequence of available transformations tested. If a given transformation $\theta_i \in \Theta$ is found to improve the local distribution of element quality metrics, such that $\mathcal{Q}(\theta_i(\mathcal{T})) > \mathcal{Q}(\mathcal{T})$, the transformation is accepted and all new elements $t_i \in \theta_i(\mathcal{T})$ are appended to the queue $\mathcal{P}$ as new active entries. An active element t_i becomes inactive when there is no transformation $\theta_i \in \Theta$ that can be successfully applied in its local neighbourhood. Inactive elements are simply removed from the queue $\mathcal{P}$. The greedy, queue-based algorithm converges to a locally optimal configuration when the queue becomes empty. Several variants of this algorithm can be constructed, based on the manner in which the priority queue $\mathcal{P}$ is maintained. In this thesis, I have investigated two options: (i) *first-come*, in which active elements are processed according to the simple linear order in which they are added to the queue, and (ii) *worst-first*, in which the queue $\mathcal{P}$ is sorted according to element quality, with the lowest quality elements processed first. The worst-first heuristic is designed to prioritise the improvement of the worst elements

in a mesh and has been found to produce superior results in practice. The worst-first criteria is adopted throughout the remainder of this work.

While the core framework of the greedy, queue-based procedure clearly generalises to different types of tessellations, including the triangular and tetrahedral types considered in this thesis, the application of the transformation catalogue must be defined explicitly for each element type. Given a 2-simplex $t_i \in \mathcal{T}$, a sequence of $\mathcal{T}_2\mathcal{T}_2$ flips are attempted about each edge $e_j \in t_i$, with the first transformation that leads to an improvement in the local mesh quality metrics accepted. Given a 3-simplex $t_i \in \mathcal{T}$, a series of edge- and face-based flips are attempted. A sequence of $\mathcal{G}_2\mathcal{G}_3$ and $\mathcal{G}_4\mathcal{G}_4$ flips are first attempted about each face $f_j \in t_i$. If such flips are unsuccessful, a sequence of $\mathcal{G}_3\mathcal{G}_2$ flips are then attempted about each edge $e_j \in t_i$. Consistent with the procedure used for 2-simplexes, the first transformation leading to an improvement in the local mesh quality metrics is accepted. In this thesis, the ordering of the edge- and face-based transformations is randomised, in an effort to prevent the algorithm becoming unnecessarily 'stuck' on local maxima. I make no claims concerning the optimality of the flip sequences presented here, though I have found that such algorithms are effective in practice. The queue-based procedure for local topological optimisation is summarised in Algorithm 6.2.1.

6.2.3 Vertex Smoothing

The second class of mesh improvement predicates investigated in this thesis is focused on the geometric optimality of the underlying tessellation. Closely following an approach rigorously developed by Freitag and Olivier-Gooch [13], Freitag and Knupp [12] and Klinger and Shewchuk [19, 20], a set of locally optimal vertex positions can be constructed via a local optimisation procedure. Given a set of vertex coordinates $X \subset \mathbb{R}^d$ and an associated simplicial tessellation $\mathcal{T}(X)$, the position of any given vertex $\mathbf{x}_i \in X$ can be optimised with respect to its local element neighbourhood. The position of a given vertex $\mathbf{x}_i$ is considered to be locally optimal when the distribution of quality metrics in the associated 'patch' $\mathcal{T}_{\mathbf{x}_i} = \{\tau \in \mathcal{T} \,|\, \tau \cap \mathbf{x}_i \neq \emptyset\}$ adjacent to the vertex $\mathbf{x}_i$ is maximised. Consistent with the hill-climbing paradigm outlined previously, I seek a maximisation of the *worst-case* quality metric in $\mathcal{T}_{\mathbf{x}_i}$ in this work. Following conventional strategies, the optimisation of all vertices $\mathbf{x}_i \in X$ in a given tessellation is sought iteratively, by incrementally applying the geometric optimisation algorithm in the neighbourhood of each vertex in turn. Such a process is repeated until further improvements in mesh quality are sufficiently small. Due to the incremental movement of vertex positions, geometrical optimisation is often known as a *vertex smoothing* procedure. The full vertex smoothing procedure is outlined in Algorithm 6.2.2.

6.2.3.1 Laplacian Smoothing

Arguably the simplest local geometric optimisation predicate is the well-known *Laplacian smoothing* procedure [9]. Given a vertex $\mathbf{x}_i \in X$ in a simplicial tessellation $\mathcal{T}(X)$, the Laplacian smoothing operation follows a simple strategy – setting the vertex $\mathbf{x}_i$ equal to the centroid of the polyhedron formed by the set of its 1-ring vertex neighbours $X_{\mathbf{x}_i} \subseteq X$,

Algorithm 6.2.2 Local Vertex Smoothing

function SMOOTHVERTEX($X, \mathcal{T}_X, \mathbf{x}_i$) ▷ {update vertex position}
  Evaluate initial vertex metric $\mathcal{Q}_{\mathbf{x}_i}$.
  if ($\mathcal{Q}_{\mathbf{x}_i} \leq \beta$) **then**
    return OPTIMISERSMOOTH($X, \mathcal{T}_X, \mathbf{x}_i$).
  else
    return LAPLACIANSMOOTH($X, \mathcal{T}_X, \mathbf{x}_i$).
  end if
end function

function LAPLACIANSMOOTH($X, \mathcal{T}_X, \mathbf{x}_i$) ▷ {Laplacian update}
  Form vertex update: $\tilde{\mathbf{x}}_i = \left(1/|X_{\mathbf{x}_i}|\right) \sum_{j=1}^{|X_{\mathbf{x}_i}|} \mathbf{x}_j$.
  Form projected update $\tilde{\mathbf{x}}_i'$ using $\tilde{\mathbf{x}}_i$.
  Evaluate updated vertex metric $\tilde{\mathcal{Q}}_{\mathbf{x}_i}'$.
  return ($\tilde{\mathcal{Q}}_{\mathbf{x}_i} \geq \mathcal{Q}_{\mathbf{x}_i}$).
end function

function OPTIMISERSMOOTH($X, \mathcal{T}_X, \mathbf{x}_i$) ▷ {optimisation-based update}
  while (Pass and $i \leq$ MAXITER) **do**
    Form gradient vectors $\mathbf{g}_j(\mathbf{x}_i)$.
    Form descent direction $\tilde{\mathbf{g}}$.
    Compute initial line-search step-size Δ^0.
    Pass = FALSE.
    while $\left((\text{not Pass}) \text{ and } \Delta^k \geq \Delta_{\min}\right)$ **do** ▷ {line-search}
      Attempt vertex update $\tilde{\mathbf{x}}_i^* = \tilde{\mathbf{x}}_i^k + \Delta^k \tilde{\mathbf{g}}$.
      Form projected update $\tilde{\mathbf{x}}_i'$ using $\tilde{\mathbf{x}}_i^*$.
      Evaluate updated vertex metric $\tilde{\mathcal{Q}}_{\mathbf{x}_i}'$.
      if ($\tilde{\mathcal{Q}}_{\mathbf{x}_i} \geq \mathcal{Q}_{\mathbf{x}_i}$) **then**
        Pass = TRUE.
      end if
      Bisect step-size $\Delta^{k+1} = \frac{1}{2}\Delta^k$. ▷ {step-size reduction}
    end while
    Update iteration counter $i = i + 1$.
  end while
end function

such that:

$$\tilde{\mathbf{x}}_i = \frac{1}{|X_{\mathbf{x}_i}|} \sum_{j=1}^{|X_{\mathbf{x}_i}|} \mathbf{x}_j \tag{6.1}$$

The Laplacian smoothing operator acts to 'smooth' an initial vertex distribution in an isotropic fashion, repositioning each vertex to the mean of its local neighbourhood. Such an action is similar to the application of a conventional (differential) Laplacian operator, giving rise to its name. Noting that the Laplacian smoothing update is not guaranteed to improve worst-case element quality, a variation known as *smart* Laplacian-smoothing [11], is employed in this thesis. Rather than simply accepting the updated vertex coordinate $\tilde{\mathbf{x}}_i$ directly, smart Laplacian smoothing is based on an evaluation of the change in local quality metrics $\mathcal{Q}_{t_i}(\mathbf{x}_i)$ for all elements $t_i \in \mathcal{T}_{\mathbf{x}_i}$. Provided that a sufficient improvement in local quality is achieved, the updated coordinate $\tilde{\mathbf{x}}_i$ is accepted. For consistency with the overall hill-climbing paradigm, I require that such updates improve the worst-case quality metric. Laplacian smoothing can be used to improve the geometric quality of a tessellation globally by simply iterating over its vertices one-by-one.

6.2.3.2 Non-Smooth Optimisation

While Laplacian smoothing is known to be effective when the initial quality of the tessellation is already reasonably high, it is typically less effective when attempting to improve general tessellations, especially in the tetrahedral case. Freitag [11], along with contributions from a number of other authors, including Olivier-Gooch [13], Knupp [12], and Jones and Plassman [10], have developed several optimisation-based approaches for vertex smoothing, designed to offer improved effectiveness and robustness. Considering the optimisation of a single vertex position $\mathbf{x}_i \in X$, the task is to find the new position $\tilde{\mathbf{x}}_i$ that leads to a maximisation of the minimum quality metric in the surrounding 1-ring neighbourhood $\mathcal{T}_{\mathbf{x}_i} \subseteq \mathcal{T}$:

$$\mathcal{Q}_{\mathbf{x}_i} = \min(\mathcal{Q}_{t_1}(\mathbf{x}_i), \mathcal{Q}_{t_2}(\mathbf{x}_i), \ldots, \mathcal{Q}_{t_n}(\mathbf{x}_i)) \tag{6.2}$$

Even when the element-wise quality function $\mathcal{Q}_{t_i}(\mathbf{x}_i)$ is sufficiently smooth (as is the case for the area- and volume-length metrics used in this thesis), the action of the min() operator implies that the vertex-based objective function $\mathcal{Q}_{\mathbf{x}_i}$ is inherently non-smooth. Following closely the methods described by Freitag and Olivier-Gooch [13], I use a non-smooth variant of the steepest descent technique to optimise (6.2). Firstly, any element(s) associated with the minimum quality metric $\mathcal{Q}_{t_k} = \mathcal{Q}_{\mathbf{x}_i}$ constitute an *active-set* $\mathcal{A}_{\mathcal{T}_i} \subseteq \mathcal{T}_{\mathbf{x}_i}$. At the k-th step of the optimisation procedure, we seek a vertex position $\mathbf{x}_i^{k+1}$ leading to an improvement in the vertex-centred metric $\mathcal{Q}_{\mathbf{x}_i}^{k+1} > \mathcal{Q}_{\mathbf{x}_i}^{k}$. Improvements in $\mathcal{Q}_{\mathbf{x}_i}$ correspond to changes in the membership of the active-set $\mathcal{A}_{\mathcal{T}_i}$. The search direction at each step is calculated via the gradient directions $\mathbf{g}_j(\mathbf{x}_i) = \partial/\partial_{\mathbf{x}_i}(\mathcal{Q}_{t_j}(\mathbf{x}_i))$ as the solution of a local quadratic programming problem:

$$\min(\tilde{\mathbf{g}}^{\mathrm{T}}\tilde{\mathbf{g}}) \text{ where } \tilde{\mathbf{g}} = \sum_{j \in \mathcal{A}_{\mathcal{T}_i}} \alpha_j \mathbf{g}_j(\mathbf{x}_i) \tag{6.3}$$

$$\text{subject to } \sum_{j \in \mathcal{A}_{\mathcal{T}_i}} \alpha_j = 1, \;\; \alpha_j \in \mathbb{R}^+ \tag{6.4}$$

Given a search vector $\tilde{\mathbf{g}}$, an update for the vertex position is calculated, setting $\tilde{\mathbf{x}}_i^{k+1} = \tilde{\mathbf{x}}_i^k + \Delta\,\tilde{\mathbf{g}}$, where Δ is a scalar step-size, $\Delta \in \mathbb{R}^+$. An estimate for the step length Δ is computed using a first-order finite-difference approximation for the change in element metrics:

$$\mathcal{Q}_j(\mathbf{x}_i + \Delta) = \mathcal{Q}_j(\mathbf{x}_i) + \Delta\,\tilde{\mathbf{g}}_j(\mathbf{x}_i) \tag{6.5}$$

Using this information, an estimate of the minimum Δ for which there is a change in membership in the active-set $\mathcal{A}_{\mathcal{T}_i}$ is computed, and is used as an initial guess for the line-search increment Δ^0. The line-search proceeds, requiring that an improvement in the vertex metric $\mathcal{Q}_{\mathbf{x}_i}$ be realised for a given Δ^k. A recursive bisection is invoked

otherwise, setting $\Delta^{k+1} = \frac{1}{2}\Delta^k$ until a successful step-size is found. If instead the maximum number of line-search iterations is exhausted, a failure is reported and local optimisation is terminated. Contrary to the original work of Freitag et al. [10, 11, 13], or the subsequent studies of Klinger and Shewchuk [19], I evaluate all gradients $\partial/\partial\mathbf{x}(\mathcal{Q}_{t_j}(\mathbf{x}_i))$ numerically using second-order accurate finite-differences, ensuring that the step-size is a small fraction of the minimum edge length adjacent to the vertex $\mathbf{x}_i$.

6.2.3.3 Constraints

Both the Laplacian- and optimisation-based geometric operators update the vertex coordinates in full-dimensional space, and do not, as a result, enforce external constraints on vertex movement. To ensure that the geometric constraints imposed by the geometry of the domain are correctly satisfied, the vertex update strategies described above are modified accordingly. In this thesis, I adopt a simple projection-based strategy, in which the Laplacian- and optimisation-based vertex updates are modified by an additional projection step, in which pending updates are projected onto the closest segment of their associated constraints. Three classes of constraint are supported in this study: (i) corner constraints, (ii) curve-based constraints, and (iii) surface-based constraints. The Laplacian- and optimisation-based vertex updates are modified to ensure that: (i) any corner vertices remain fixed, (ii) any curve vertices are projected onto the curve, but are allowed to 'slide' along its length, and (iii) any surface vertices are projected onto the surface, but are allowed to 'slide' over its area. Given a constrained vertex $\mathbf{x}_i \in X$, an unconstrained update $\tilde{\mathbf{x}}_i$ is first calculated, based on either the Laplacian smoothing or non-smooth optimisation predicates. A projected update $\tilde{\mathbf{x}}'_i$ is then calculated and the vertex-centred quality metric $\mathcal{Q}'_{\mathbf{x}_i}$ evaluated at this projected position. If there is an improvement in the vertex-centred metric $\mathcal{Q}'_{\mathbf{x}_i}$, the projected update $\tilde{\mathbf{x}}'_i$ is accepted. Such a method is equivalent to taking the component of the unconstrained update $\tilde{\mathbf{x}}_i$ aligned with the constraining curve/surface.

6.2.3.4 Combined Smoothing

While optimisation-based vertex updates are known to be significantly more effective than Laplacian smoothing when attempting to improve the quality of the worst elements in a given tessellation, they are also substantially more computationally expensive. To ameliorate this extra cost, Freitag and Olivier-Gooch [11, 13] advocate for the use of a hybrid *combined* smoothing process, in which some combination of Laplacian smoothing and optimisation-based improvement are applied concurrently. In this thesis, I make use of the 'first-method' of Freitag [11], where, given a vertex $\mathbf{x}_i$, I select the smoothing scheme based on the initial value of the vertex-centred quality metric $\mathcal{Q}_{\mathbf{x}_i}$. If $\mathcal{Q}_{\mathbf{x}_i}$ exceeds a user-defined smoothing threshold β, a Laplacian smoothing update is attempted. Otherwise, if $\mathcal{Q}_{\mathbf{x}_i} \leq \beta$, the optimisation-based method is used. Such a procedure is designed to reserve the expensive optimisation-based updates for *difficult* vertices that are not improved by the relatively cheap Laplacian smoothing operation. The smoothing threshold β, ensures that optimisation-based updates are only attempted for vertices in

Algorithm 6.2.3 Local Vertex Insertion

```
1: function RefineEdge(X, t_j)                          ▷ {longest edge refinement}
2:     Find longest edge e_max ∈ t_j.
3:     Form new Steiner vertex c = e_mid.
4:     if (TestFlipEmid(X, t_j, c)) then
5:         Update X ← c.                                ▷ {update vertices/topology}
6:         Update T ← H'.
7:         return True.
8:     end if
9: end function
```

the neighbourhood of sufficiently low-quality elements.

6.2.4 Vertex Insertion

The final class of mesh improvement predicates investigated in this thesis are based on a sequence of *refinement* templates, designed to improve local mesh quality through the introduction of new Steiner vertices. The use of refinement in the context of mesh improvement is an idea introduced by Klinger and Shewchuk [19, 20], where they focus on the development of techniques for the improvement of tetrahedral meshes. Given a low-quality element $t_i \in \mathcal{T}$, Klinger and Shewchuk seek to construct an optimal *cavity* $\mathcal{H} \subseteq \mathcal{T}$ for a new Steiner vertex $\mathbf{c}$. Through a re-triangulation of the cavity $\mathcal{H}$ about the new vertex $\mathbf{c}$, an improvement in the local element quality metrics $\mathcal{Q}_{t_j}(\mathbf{x})$ is sought, consistent with the discussions presented previously. In practice, this task is known to be difficult. In general, it is not known how to efficiently compute either the position of the vertex $\mathbf{c}$, or the topology of the associated cavity $\mathcal{H}$ to guarantee a maximisation of the worst-case metric $\min(\mathcal{Q}_{t_j}(\mathbf{c}))$ for all t_j in the re-triangulated cavity $\mathcal{H}'$.

In light of these difficulties, Klinger and Shewchuk instead focus on the construction of an effective heuristic scheme. In [20], they develop a *composite* refinement operation for tetrahedral meshes. Given an element $t_i \in \mathcal{T}$ marked for refinement, their composite strategy performs a *sequence* of operations, in which: (i) a new Steiner vertex $\mathbf{c}$ is first positioned about the circumcentre of t_i, (ii) a pseudo-optimal star-shaped cavity $\mathcal{H} \subseteq \mathcal{T}$ is induced in the neighbourhood of t_j via a greedy breath-first search of the underlying adjacency graph, (iii) the cavity is re-triangulated, forming a new sub-mesh $\mathcal{H}'$, and (iv) a range of smoothing and topological improvement operations are attempted in the neighbourhood of $\mathcal{H}'$. This composite operation is accepted if there is an improvement in the worst-case quality metrics $\mathcal{Q}_{t_j}(\mathbf{x})$ for all t_j in the new sub-mesh $\mathcal{H}'$. If no such improvement is realised, the full sequence of operations associated with the operation are unwound, via a *roll-back* mechanism. While Klinger and Shewchuk report that such composite refinement operations are effective in practice, significantly improving the quality of even very low-quality tessellations, they also note that such operations impose significant computational costs – contributing as much as 90% of the total work associated with their full mesh optimisation program [20].

In this thesis, I pursue a simpler and more cost effective strategy. Recall firstly that the planar, surface and volumetric Frontal-Delaunay meshing algorithms presented in Chapters 3, 4 and 5 are designed to guarantee the generation of high-quality tessella-

tions with bounded radius-edge ratios $\rho(\tau)$. Given that the input is expected to be of reasonable quality, I introduce a simple *edge-centred* refinement template. Following the strategy of Klinger and Shewchuk described previously, given an edge $e \in \mathcal{T}$ marked for refinement, the edge-refinement predicate attempts the following sequence of operations: (i) a new Steiner vertex $\mathbf{c} \in e$ is selected, (ii) an associated star-shaped cavity $\mathcal{H} \subseteq \mathcal{T}$ is found, and (iii) the cavity is re-triangulated, forming the new sub-mesh $\mathcal{H}'$. In this work, I do not require that the cavity $\mathcal{H}$ be optimal, and simply select the set of elements in the 1-ring neighbourhood of the edge e. Consistent with the set of optimisation predicates discussed previously, the edge-refinement predicate is successful if there is an improvement in the worst-case quality metrics $\mathcal{Q}_{t_j}(\mathbf{x})$ for all t_j in the new sub-mesh $\mathcal{H}'$.

Based on practical experience with a range of mesh optimisation problems, it was found that elements lying on the boundary of the tessellation containing a long edge were often resistant to both the vertex smoothing and topological transformation predicates introduced previously. These findings are consistent with those of Klinger and Shewchuk in [20]. As a result, a simple *edge-bisection* refinement scheme is adopted in this work in which the new Steiner vertex $\mathbf{c} \in e$ is positioned at the edge midpoint $\mathbf{e}_{\text{mid}}$.

6.3 Mesh Improvement Framework

I assemble a new mesh optimisation franework, known as `JITTERBUG`, by combining the various geometrical, topological and insertion-based optimisation predicates introduced throughout Section 6.2. Existing mesh improvement algorithms typically consist of the iterative application of a set of basic optimisation predicates according to a predefined *schedule*. In the early work of Freitag and Olivier-Gooch [13], such optimisation schedules were typically *static*, with the total mesh improvement procedure comprised of a fixed sequence of operations. While this type of approach leads to the development of relatively straightforward algorithms, it does not cater for the specific character of a given tessellation. As a result, such methods are known to be guilty of performing either too much or too little optimisation, leading to reductions in computational efficiency or sub-optimal optimisation performance, respectively. In their tetrahedral mesh improvement program `STELLAR`, Klinger and Shewchuk [19, 20] instead investigate the use of a *dynamic* optimisation schedule, in which the application of specific local optimisation predicates is adapted based on the convergence characteristics of the overall process. Such a strategy has been shown to result in very high levels of optimisation performance for a wide range of input meshes. Unfortunately, despite these successes, the `STELLAR` program is known to be still very computational expensive in practice, often requiring some orders of magnitude more computational effort to improve a mesh than was initially used to generate it [20].

In this thesis, I develop an adaptive optimisation schedule, designed to incorporate the flexibility and performance of the scheme previously introduced by Klinger and Shewchuk, while also improving on its computational performance. Additionally, I focus on the development of a general mesh improvement framework, catering to planar, surface and volumetric tessellations, rather than supporting the optimisation of tetrahedral meshes

Algorithm 6.3.1 Mesh Improvement Schedule

```
1: function ImproveMesh(X, T, γ)
2:     Call OptimTopology(X, T).                    ▷ {form locally optimal topology}
3:     Initialise active vertex set A_X ← X.
4:     while (A_X ≠ ∅) do
5:         for (all x_i ∈ A_X) do                   ▷ {perform refinement pass}
6:             Form 1-ring neighbourhood T_X ← Ring(x_i).
7:             for (all t_i ∈ T_X) do
8:                 Call RefineEdge(X, t_i).
9:             end for
10:        end for
11:        Initialise Q_X ← ∅.
12:        for (all x_i ∈ A_X) do                   ▷ {perform smoothing pass}
13:            Form 1-ring neighbourhood T_X ← Ring(x_i).
14:            if (SmoothVertex(X, T_X, x_i)) then
15:                Call OptimTopology(X, T_X).
16:                Store active vertex Q_X ← x_i.
17:            end if
18:        end for
19:        Initialise A_X ← ∅.
20:        for (all x_i ∈ Q_X) do                   ▷ {update active vertex set}
21:            Form local length measure ‖e‖.
22:            if (Δx_i ≥ γ ‖e‖) then
23:                Update active set A_X ← x_i.
24:                Inflate A_X about x_i.
25:            end if
26:        end for
27:    end while
28: end function
```

```
1: function Ring(x_i)                               ▷ {assemble 1-ring neighbourhood}
2:     return R_i = {τ_i ∈ T | τ_i ⋂ x_i ≠ ∅}
3: end function
```

only. The new framework is based on a *priority-driven* optimisation schedule, in which optimisation is focused on the worst elements in the tessellation.

Given a simplicial tessellation $\mathcal{T}(X)$, the new mesh improvement algorithm first constructs a locally optimal topology $\mathcal{T}'$ by applying the greedy topological optimisation procedure outlined in Algorithm 6.2.1 exhaustively to all elements $t_i \in \mathcal{T}$. This topological optimality is maintained as an invariant throughout the mesh improvement procedure. Prior to the main iterative optimisation loop, all vertices $\mathbf{x}_i \in X$ are initially added to the 'active' set $\mathcal{A}_X$, marking them for optimisation. Within the main loop of the algorithm, an initial refinement sweep is undertaken, in which all elements adjacent to a vertex in the active set $\mathcal{A}_X$ are considered for refinement-based optimisation, as per the templates outlined in Section 6.2. Upon successful refinement, any new vertices are pushed onto the active set $\mathcal{A}_X$. Following completion of the refinement sweep, a combined geometrical/topological optimisation pass is initiated, updating all entities in the neighbourhood of any vertex $\mathbf{x}_i \in \mathcal{A}_X$. The active set $\mathcal{A}_X$ is traversed in-order. Given a specific active vertex $\mathbf{x}_i \in \mathcal{A}_X$ a vertex smoothing operation is first attempted, according to the local geometrical optimisation process outlined in Section 6.2. If the vertex smoothing is successful, it is immediately followed by a topological operation, in which the greedy local topological optimisation procedure described in Section 6.2 is initiated from the set of elements $\mathcal{T}_{\mathbf{x}_i} = \{t_i \in \mathcal{T} \,|\, t_i \cap \mathbf{x}_i \neq \emptyset\}$ adjacent to the vertex $\mathbf{x}_i$. Such a

process ensures that the tessellation remains locally topologically optimal following any geometrical updates to the vertex positions. Any vertices that are successfully smoothed are pushed onto a linear queue $Q_X \leftarrow \mathbf{x}_i$.

Following the application of the optimisation predicates, the active vertex set $\mathcal{A}_X$ is re-calculated for the next outer iteration. All vertices $\mathbf{x}_i \in Q_X$ are considered, and are pushed onto the active set $\mathcal{A}_X$ if their movement over the current iteration is sufficiently large, such that $\Delta\mathbf{x}_i \geq \gamma \overline{\|\mathbf{e}\|}$, where $\overline{\|\mathbf{e}\|}$ is a measure of mean edge length adjacent to the vertex $\mathbf{x}_i$ and γ is a user-specified tolerance. Such a constraint acts to remove vertices from subsequent outer iterations as approach a converged state. Scaling the movement threshold based on a measure of mean adjacent edge length $\overline{\|\mathbf{e}\|}$ ensures that the convergence criteria is tailored to the local distribution of mesh size in the neighbourhood of each vertex. Following the initial allocation of vertices onto the active set, $\mathcal{A}_X$ is *inflated* to also include any vertices in the 1-ring neighbourhood of those already present. Such an operation achieves a degree of *adaptivity* in the mesh improvement program, allowing vertices that have previously become inactive to be placed back onto the active set $\mathcal{A}_X$, as a result of movement within their local neighbourhood. Importantly, such an adaptive strategy seeks to minimise computational expense by attempting to only optimise about unconverged subsets of the tessellation. The mesh improvement program terminates when the active set $\mathcal{A}_X$ becomes empty. The full algorithm is summarised in Algorithm 6.3.1.

6.4 Improvement of Planar Meshes

The performance of the new mesh optimisation framework `JITTERBUG` for planar mesh improvement was investigated experimentally, being used to optimise a variety of planar meshes generated using the Frontal-Delaunay mesh generator presented in Chapter 3. Ten benchmark problems of varying size and complexity were tested, with problems covering a range of application areas, including geospatial processing, computational fluid dynamics, and solution adaptive meshing. Both unoptimised and optimised meshes for each test case are shown in Figure 6.4, in which elements are coloured according to shape quality. Specifically, elements with plane-angles satisfying $\theta(\tau) < 12.50°$ and $\theta(\tau) > 155.0°$ are shaded red, those with plane-angles satisfying $12.50° \leq \theta(\tau) < 25.00°$ or $130.0° \leq \theta(\tau) < 155.0°$ are shaded yellow, and those with plane-angles satisfying $25.00° \leq \theta(\tau) < 37.50°$ or $105.0° \leq \theta(\tau) < 130.0°$ are shaded green. Gray elements are of better quality. Additionally, histograms of element area-length $a(\tau)$ and plane-angle $\theta(\tau)$ measures are shown in each case, indicating mean and minimum area-length ratios $\overline{a(\tau)}$, $a(\tau)_{\text{min}}$, and minimum and maximum plane angle bounds $\theta(\tau)_{\text{min}}$, $\theta(\tau)_{\text{max}}$. Note that a number of test-cases include small acute angles, seen as outlying $\theta(\tau)_{\text{min}}$ or $\theta(\tau)_{\text{max}}$ values. The optimisation runs were completed using a vertex movement tolerance $\gamma = 1 \times 10^{-3}$ and a vertex smoothing threshold $\beta = 0.9$. The planar mesh improvement program was implemented in C++ and compiled as a 64-bit executable.

Analysis of Figure 6.4 shows that the new `JITTERBUG` optimisation program is effective in practice, significantly improving the quality of all meshes in the test set. Specifically,

Figure 6.4: Planar optimisation results for the AIRFOIL and LAKE meshes. Columns show un-optimised input (left) and optimised output (right). Optimisation was performed using a convergence tolerance $\gamma = 1 \times 10^{-3}$ and a smoothing threshold $\beta = 0.9$. Elements are coloured according to the local distribution of plane-angles $\theta(\tau)$, with elements satisfying $\theta(\tau) < 12.50°$ and $\theta(\tau) > 155.0°$ shaded red, those satisfying $12.50° \leq \theta(\tau) < 25.00°$ or $130.0° \leq \theta(\tau) < 155.0°$ shaded yellow, and those satisfying $25.00° \leq \theta(\tau) < 37.50°$ or $105.0° \leq \theta(\tau) < 130.0°$ shaded green. Gray elements are of better quality. Normalised histograms of element area-length $a(\tau)$ and element plane angle $\theta(\tau)$ are also included, in addition to statistics for the total runtime $t(s)$ and required outer-iterations for the optimisation program.

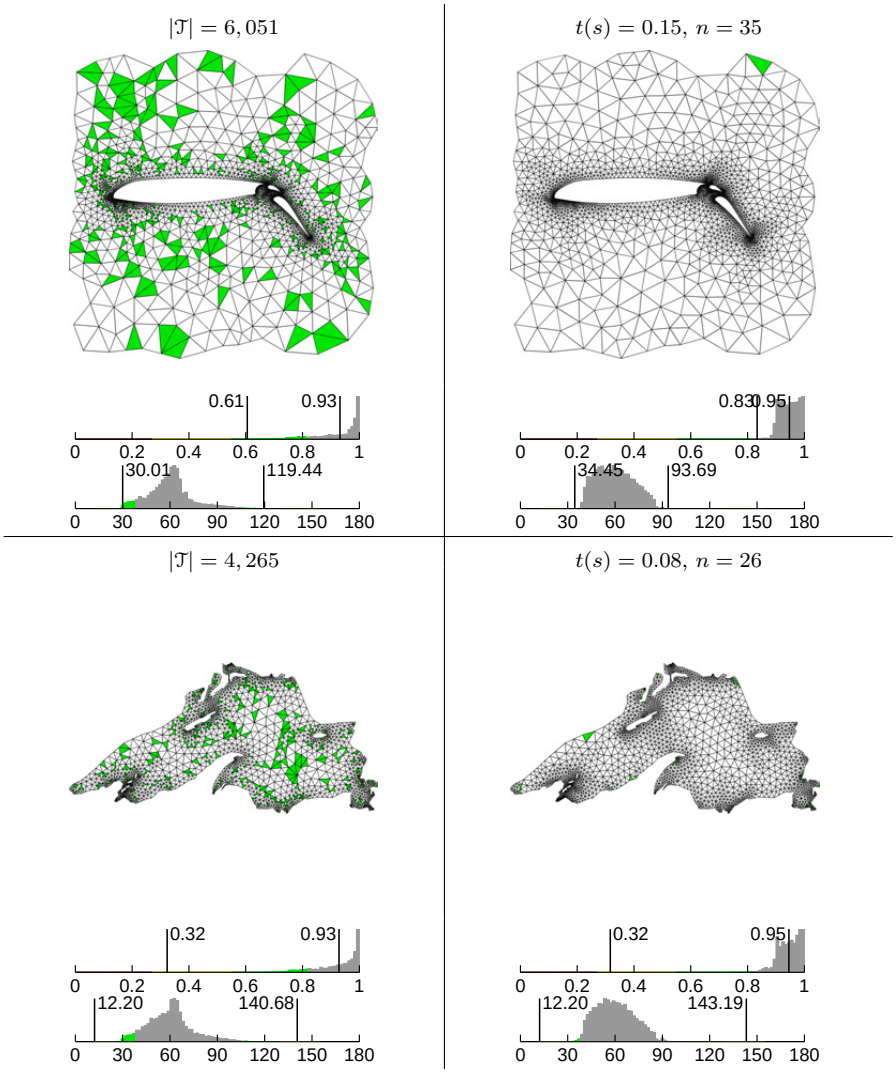


Figure 6.5: Continuation of Figure 6.4, showing comparative results for the NEWZEALAND, COASTLINE and ISLANDS test problems.

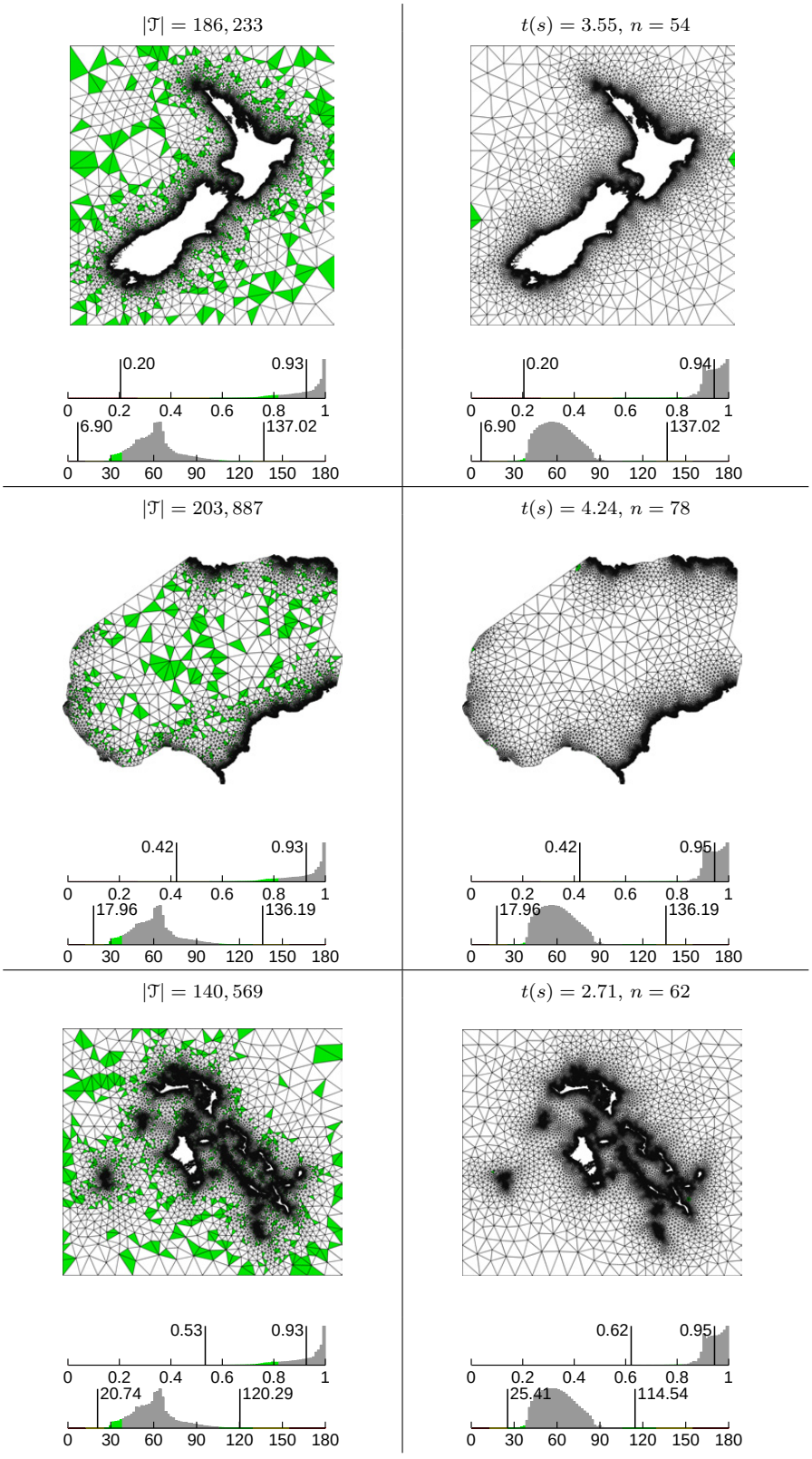


Figure 6.6: Continuation of Figure 6.4, showing comparative results for the AUSTRALIA, GREENLAND and RIVER test problems.

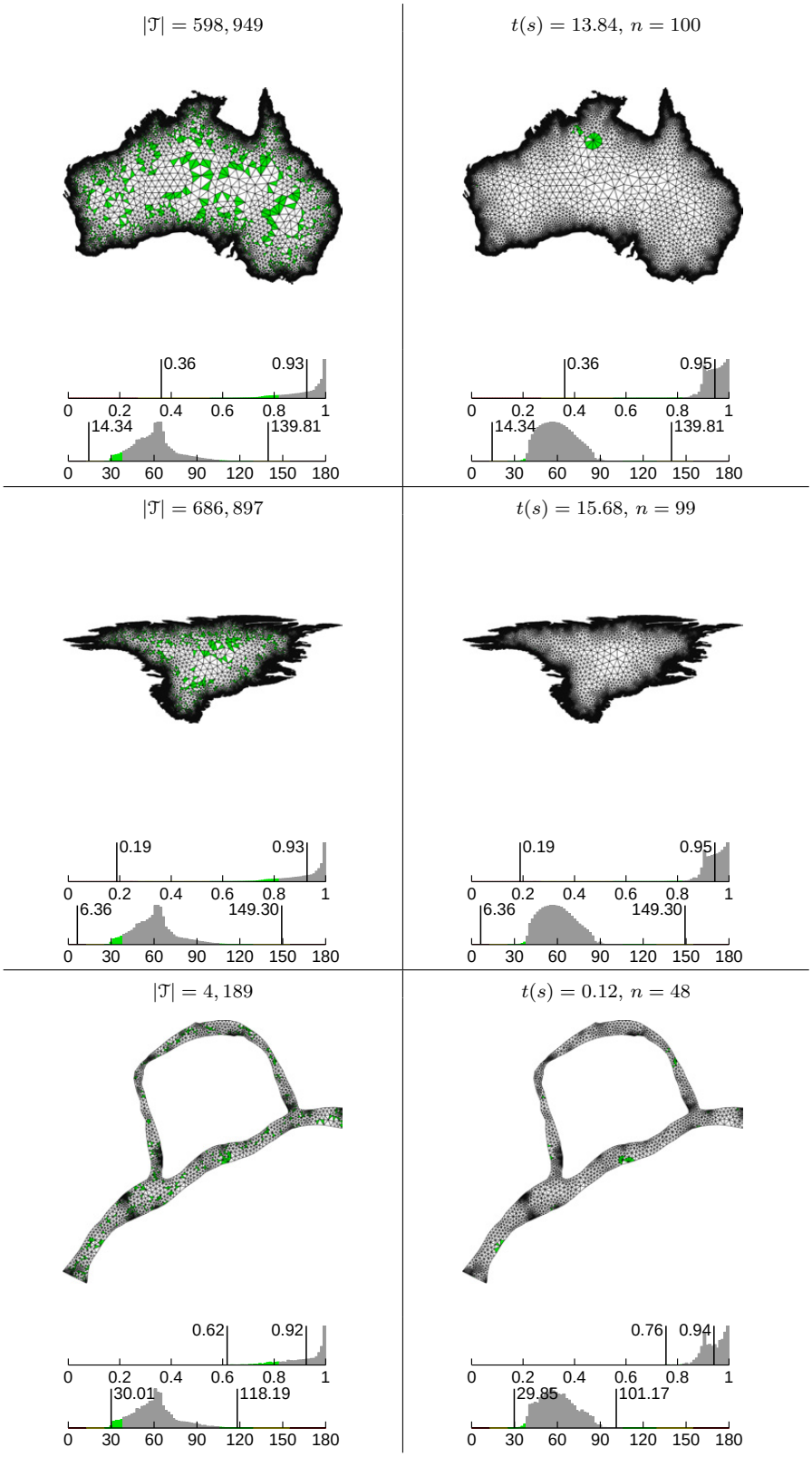
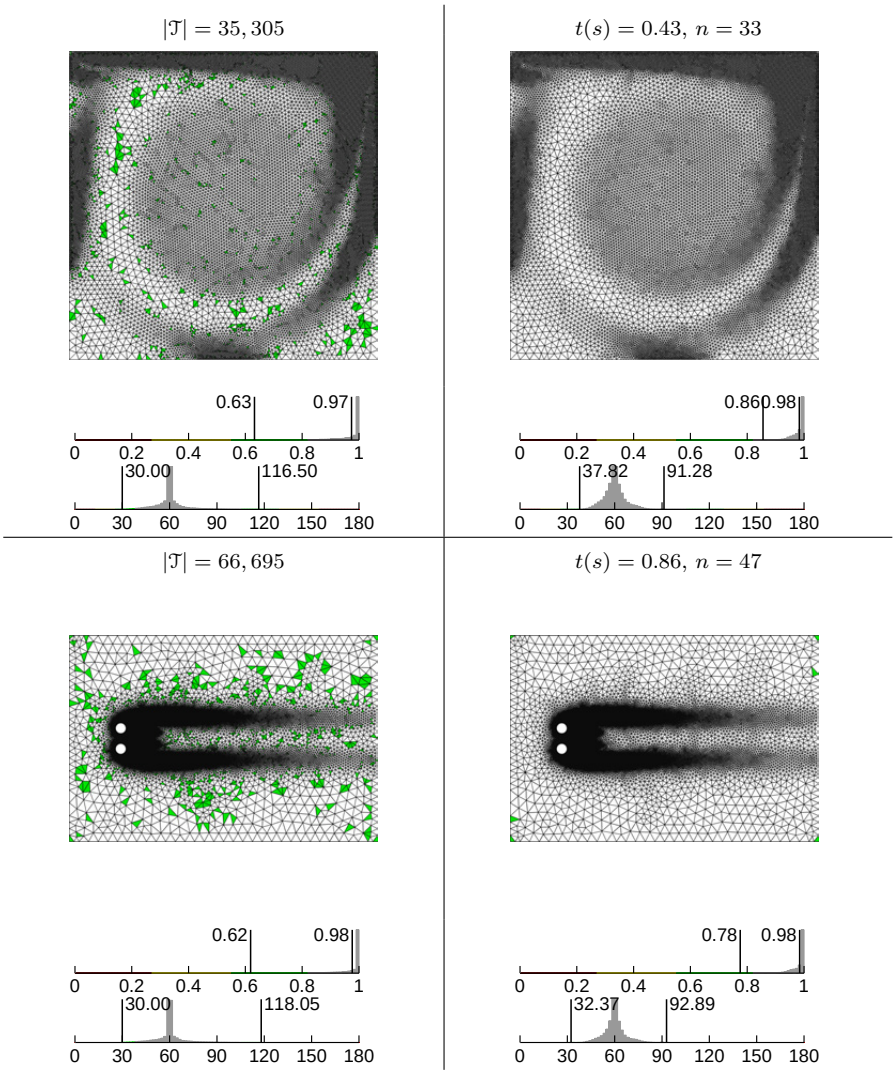


Figure 6.7: Continuation of Figure 6.4, showing comparative results for the CAVITY and CYLINDER test problems.



the mean element area-length ratios $\overline{a(\tau)}$ are seen to improve for all test cases, while the minimum are-length ratio $a(\tau)_{\min}$ are noted to improve for all tests without small constraining angles in the geometry. The optimisation process is seen to alter the shape of the $\theta(\tau)$ distribution significantly in all test cases, with a general 'fattening' typically observed due to the improvement in the shape of elements initially containing small or large plane angles. In some cases, such as the AIRFOIL, CAVITY and CYLINDER problems, the minimum and maximum plane angles, $\theta(\tau)_{\min}$ and $\theta(\tau)_{\max}$, are seen to improve beyond the maximum theoretical levels achievable using refinement-based mesh generation alone. Note also that the optimisation procedure is seen to significantly reduce the percentage of low quality elements (elements coloured red, yellow or green) present in the optimised meshes.

Despite generally impressive optimisation performance, note that a slight reduction in $\theta(\tau)_{\min}$ is observed for the RIVER test case. Initially, this behaviour may appear to

violate the hill-climbing nature of the optimisation strategy, but it must be remembered that it is the element area-length ratios $a(\tau)$ and not the plane angle measures $\theta(\tau)$ used to drive the optimisation. Noting that the element area-length $a(\tau)$ and plane angle $\theta(\tau)$ measures are related in a non-linear fashion, it is clear that a monotonic increase in $a(\tau)$ is not necessarily accompanied by similar behaviour in $\theta(\tau)$. In practice, such occurrences appear to be rare, with an optimisation of $a(\tau)$ leading to improvements in the distribution of $\theta(\tau)$ in all other cases. In the RIVER benchmark specifically, note that a significant improvement in $a(\tau)_{\text{min}}$ is still accomplished via the optimisation process.

Analysis of the optimisation runtime and iteration counts included in Figure 6.4 show that the new planar mesh improvement framework is efficient in practice. Comparisons of the run-times with those of the Frontal-Delaunay mesh generator used to create the initial unoptimised tessellations show that the optimisation process is typically a factor of 2–3 times slower. Such performance is roughly consistent across the full set of problems in the test set. These results suggest that, consistent with the underlying Delaunay-based meshing program, the planar mesh optimiser appears to achieve pseudo-linear scaling in practice, confirming that such a method is appropriate for large-scale problems. In this study, it is shown that complex test problems containing 100,000's of elements can be meshed and optimised in a matter of seconds.

6.5 Improvement of Surface Meshes

The performance of the new mesh optimisation framework `JITTERBUG` for the optimisation of surface tessellations was investigated experimentally, being used to optimise a variety of surface meshes generated using the Frontal-Delaunay mesh generator presented in Chapter 4. Fifteen benchmark problems of varying size and complexity were tested, with problems covering a range of application areas, including computer graphics, computational engineering, and medical imaging. Both unoptimised and optimised meshes for each test case are shown in Figure 6.8, in which elements are coloured according to shape quality. Specifically, elements with plane-angles satisfying $\theta(f) < 12.50^\circ$ and $\theta(f) > 155.0^\circ$ are shaded red, those satisfying $12.50^\circ \leq \theta(f) < 25.00^\circ$ or $130.0^\circ \leq \theta(f) < 155.0^\circ$ are shaded yellow, and those satisfying $25.00^\circ \leq \theta(f) < 37.50^\circ$ or $105.0^\circ \leq \theta(f) < 130.0^\circ$ are shaded green. Gray elements are of better quality. Additionally, histograms of element area-length $a(f)$ and plane angle $\theta(f)$ measures are shown in each case, indicating mean and minimum area-length ratios $\overline{a(f)}$, $a(f)_{\text{min}}$, and minimum and maximum plane angle bounds $\theta(f)_{\text{min}}$, $\theta(f)_{\text{max}}$. The optimisation runs were completed using a vertex movement tolerance $\gamma = 1 \times 10^{-3}$ and a vertex smoothing threshold $\beta = 0.9$. The surface mesh improvement program was implemented in C++ and compiled as a 64-bit executable.

Analysis of Figure 6.8 shows that the new surface mesh improvement program is effective in practice, significantly improving the quality of all meshes in the test set. Specifically, both the mean and minimum element area-length ratios $\overline{a(f)}$ are seen to improve for all test cases. The optimisation process is also seen to alter the shape of the $\theta(f)$ distribution significantly in all test cases, with a general 'fattening' typically

Figure 6.8: Surface optimisation results for the SPHERE and ELLIPSOID meshes. Columns show unoptimised input (centre) and optimised output (right). Optimisation was performed using a convergence tolerance $\gamma = 1 \times 10^{-3}$ and a smoothing threshold $\beta = 0.9$. Elements are coloured according to the distribution of plane-angles $\theta(f)$, with those elements satisfying $\theta(f) < 12.50°$ and $\theta(f) > 155.0°$ coloured red, those elements satisfying $12.50° \leq \theta(f) < 25.00°$ or $130.0° \leq \theta(f) < 155.0°$ coloured yellow, and those elements satisfying $25.00° \leq \theta(f) < 37.50°$ or $105.0° \leq \theta(f) < 130.0°$ coloured green. Gray elements are of better quality. Normalised histograms of element area-length $a(f)$ and element plane angle $\theta(f)$ are also included, in addition to statistics for the total runtime $t(s)$ and required outer-iterations for the optimisation program.

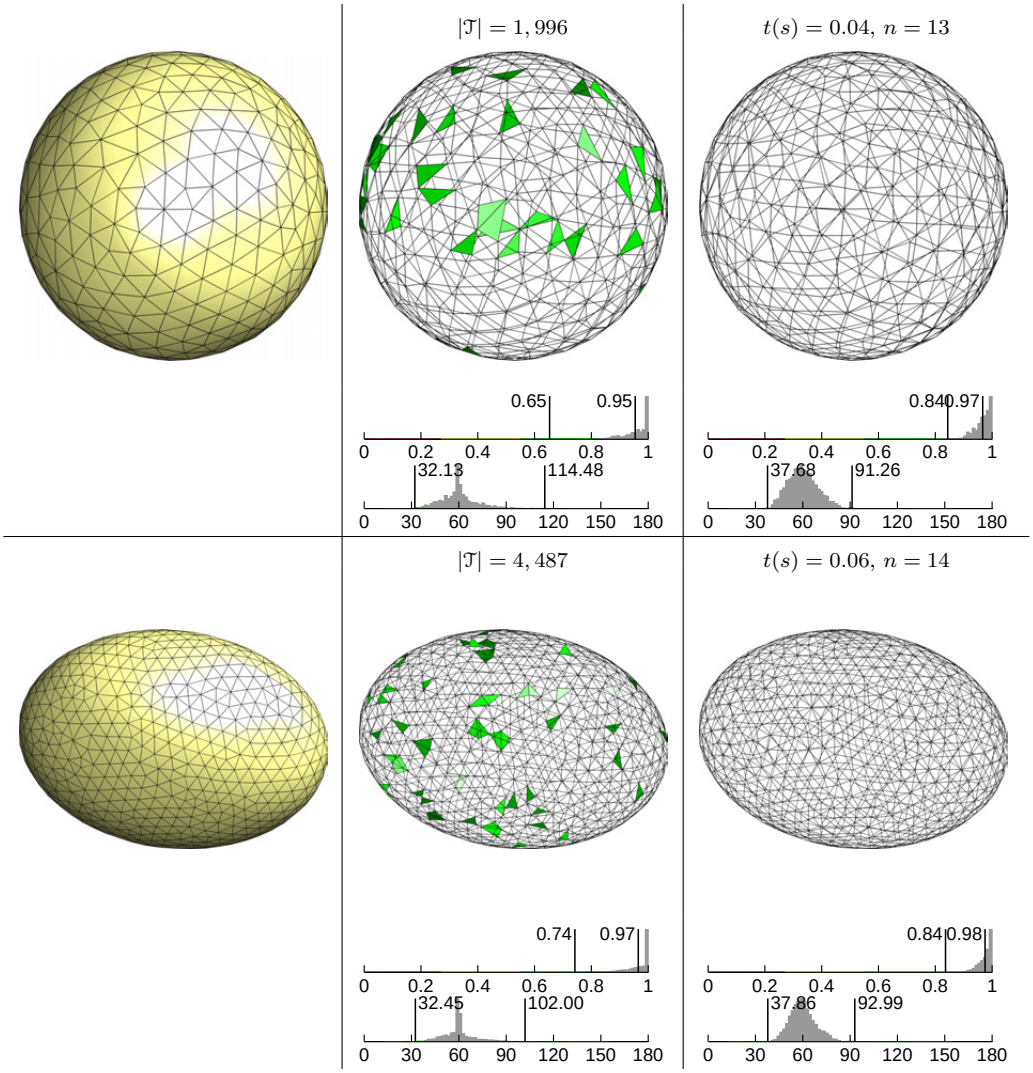


Figure 6.9: Continuation of Figure 6.8, showing comparative results for the FEMUR, HIP and VERTEBRA test problems.

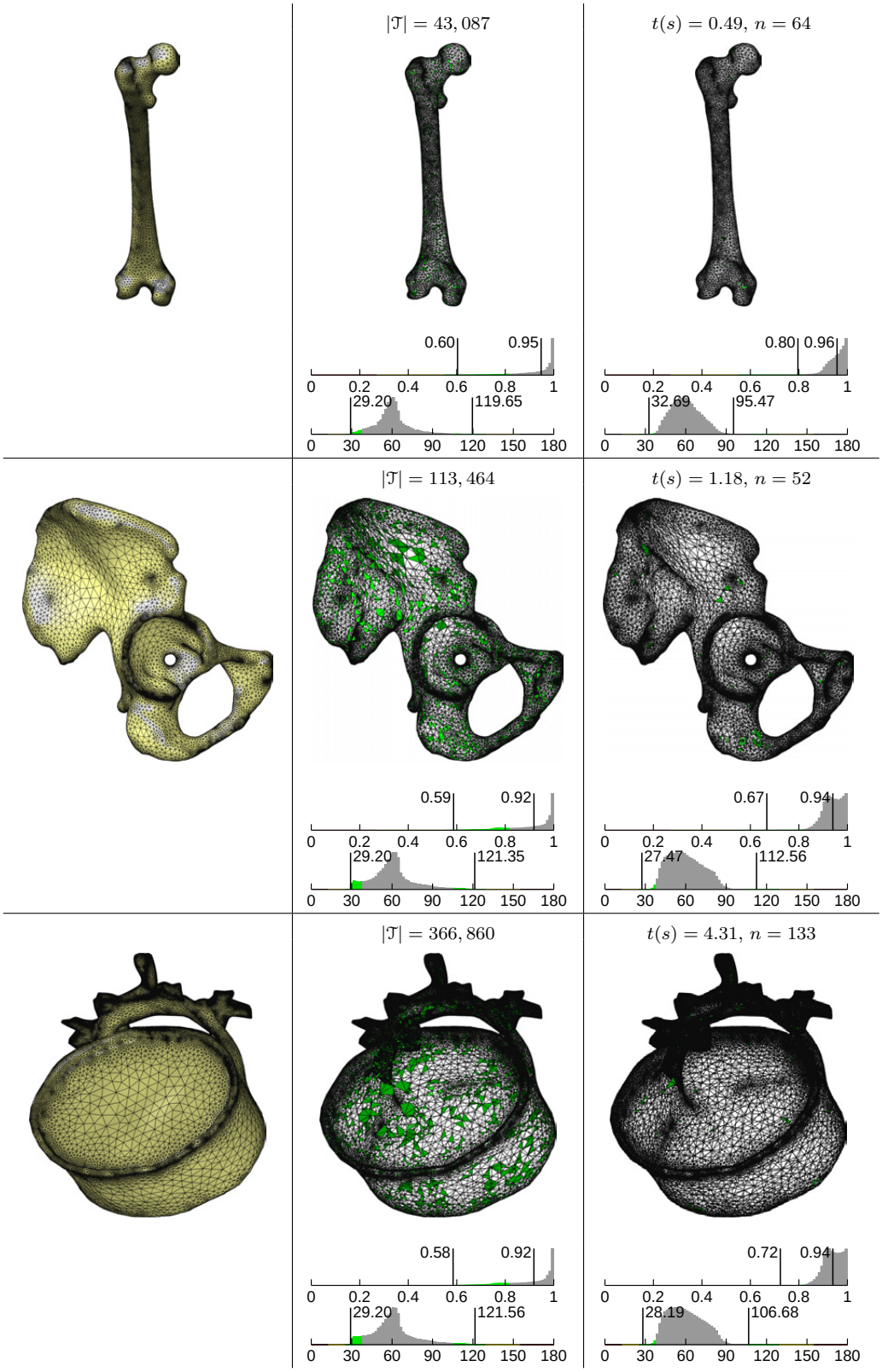


Figure 6.10: Continuation of Figure 6.8, showing comparative results for the BLADE, ROCKER and WOODTHINKER test problems.

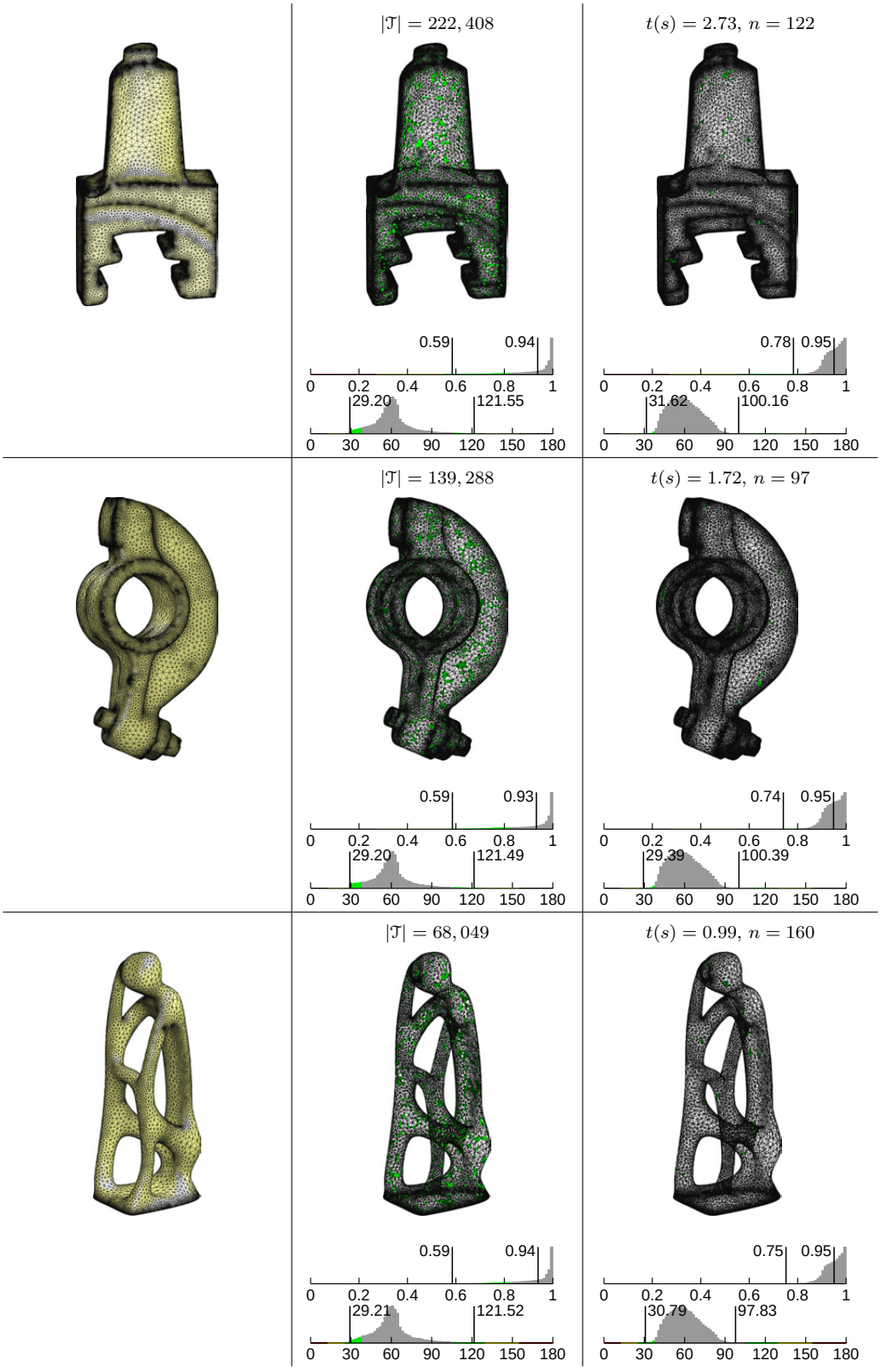


Figure 6.11: Continuation of Figure 6.8, showing comparative results for the HAND, SPIRAL 1 and SPIRAL 2 test problems.

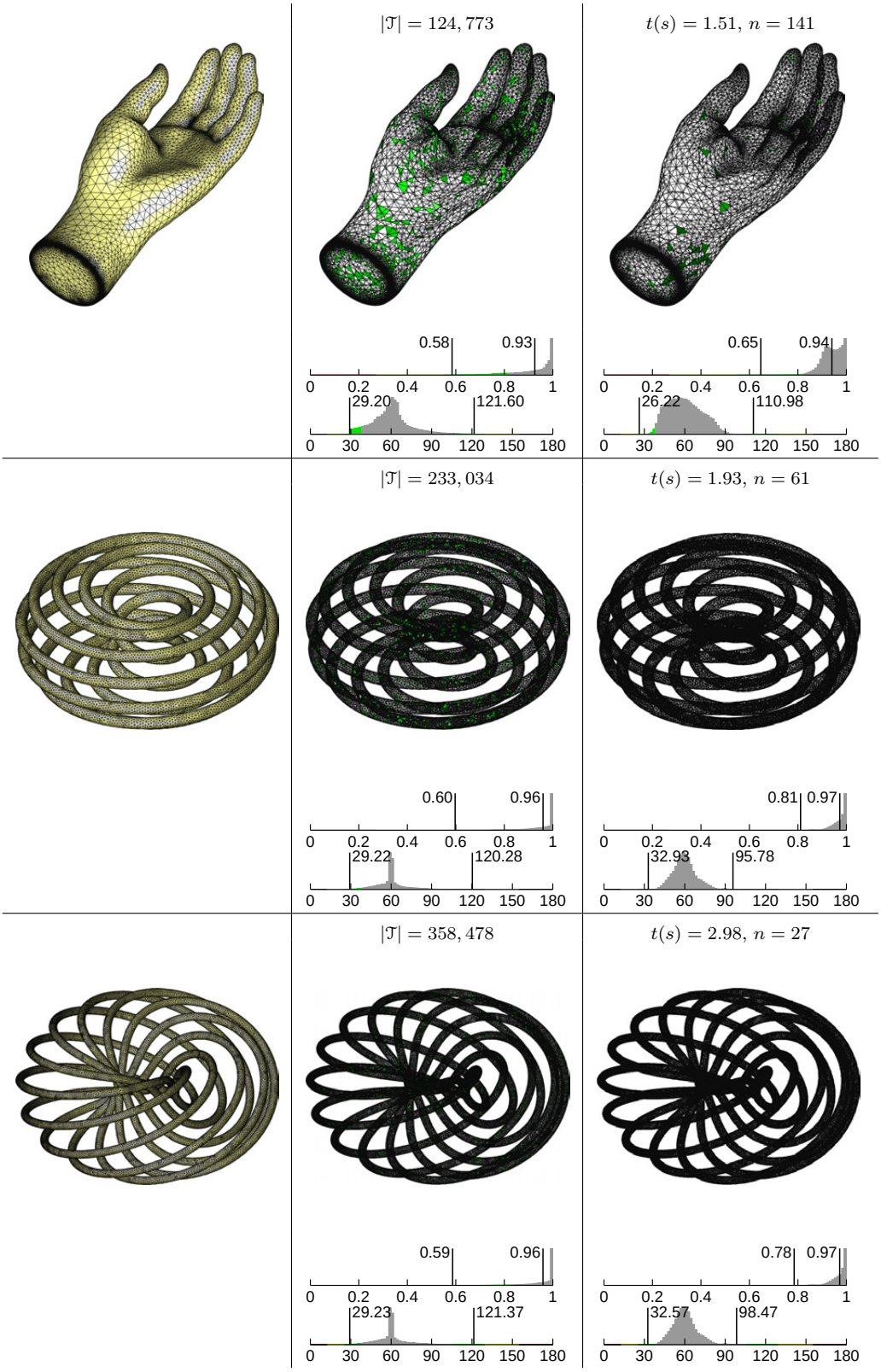


Figure 6.12: Continuation of Figure 6.8, showing comparative results for the BUNNY, BIMBA and VENUS test problems.

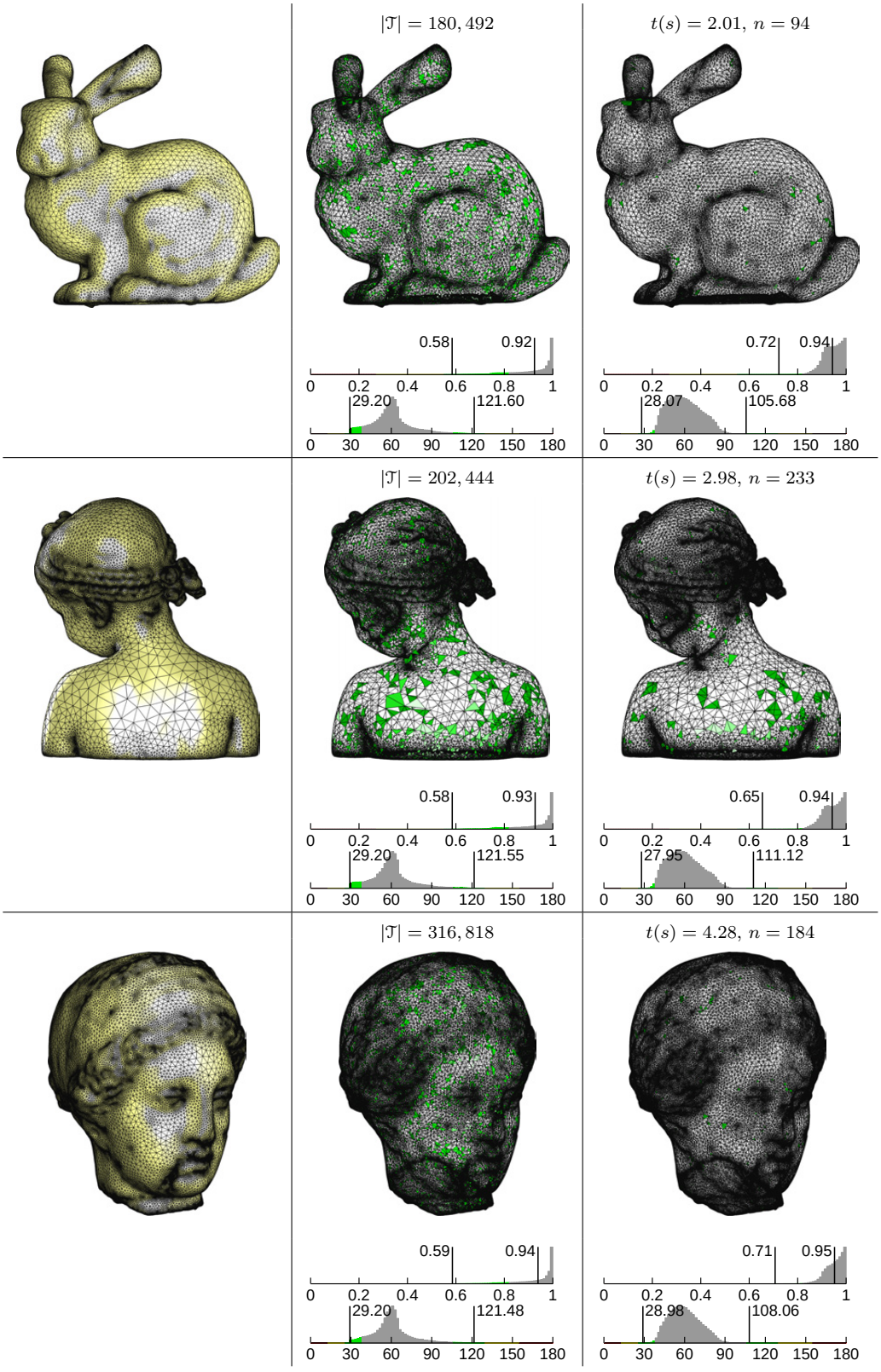
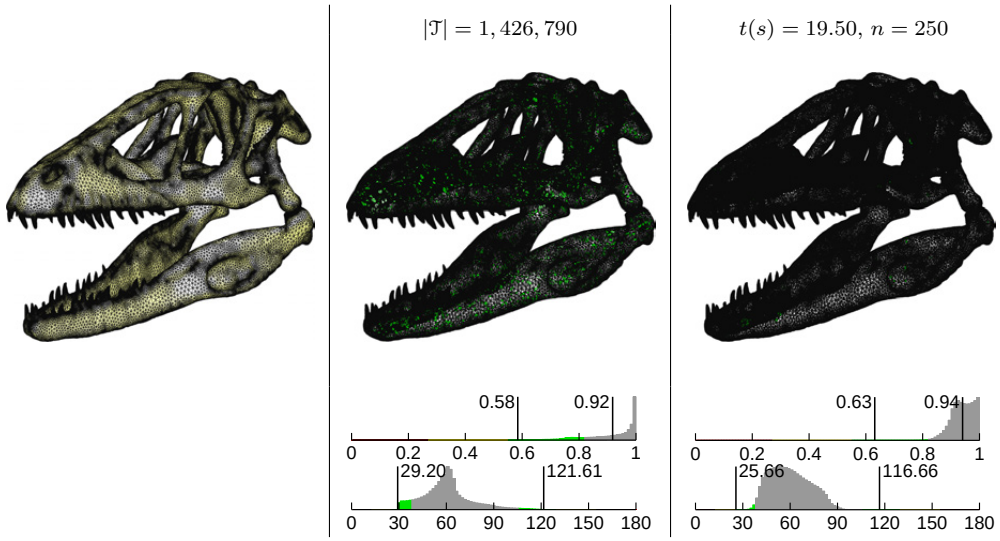


Figure 6.13: Continuation of Figure 6.8, showing comparative results for the DINOSAUR test problem.



observed due to the improvement in the shape of elements initially containing small or large plane angles. In some cases, such as the SPHERE, ELLIPSOID, FEMUR, BLADE and SPIRAL problems, the minimum and maximum plane angles, $\theta(f)_{\min}$ and $\theta(f)_{\max}$, are seen to improve beyond the maximum theoretical levels achievable using refinement-based mesh generation alone. Note also that the optimisation procedure is seen to significantly reduce the percentage of low quality elements (elements coloured red, yellow or green) in the final optimised meshes.

Consistent with the behaviour of the planar mesh improvement program, despite generally impressive optimisation performance, a slight reduction in $\theta(f)_{\min}$ is observed for a number of test-cases, including the HIP, VERTEBRA, HAND, BUNNY, BIMBA, VENUS and DINOSAUR problems. As per the discussions presented previously concerning the planar case, this behaviour does not, in fact, violate the hill-climbing nature of the optimisation strategy, but is instead due to the non-linear relationship between the element area-length $a(f)$ and plane angle $\theta(f)$ measures. Inspection of the resulting distributions confirm that the minimum element area-length ratio is increased for all test problems in the set. While an optimisation of $a(f)$ is seen to improve the distribution of $\theta(f)$ in the majority of cases, compared to the planar optimisation results presented previously, slight reductions in $\theta(f)_{\min}$ appears to be a more prevalent occurrence. The mechanism for this behaviour is not currently understood. Regardless of slight reductions in $\theta(f)_{\min}$, the surface optimisation procedure is found to result in a significant improvement in $a(f)_{\min}$ in all cases.

Analysis of the optimisation runtime and iteration counts included in Figure 6.8 show that the new surface mesh improvement framework is efficient in practice. Comparisons of the run-times with those of the Frontal-Delaunay mesh generator used to create the initial unoptimised tessellations show that the optimisation process is typically a factor of 3–4 times faster. Such performance is roughly consistent across the full set of problems in

the test set. These results suggest that, consistent with the underlying Delaunay-based meshing program, the surface mesh optimiser appears to achieve pseudo-linear scaling in practice, confirming that such a method is appropriate for large-scale problems. In this study, it is shown that complex test problems containing 100,000's of elements can be meshed and optimised in a matter of seconds.

6.6 Improvement of Tetrahedral Meshes

The performance of the new mesh optimisation framework `JITTERBUG` for the optimisation of volumetric tessellations was investigated experimentally, being used to optimise a variety of volumetric meshes generated using the Frontal-Delaunay mesh generator presented in Chapter 5 in addition to examples from external sources. Twenty-seven benchmark problems of varying size and complexity were tested, with problems covering a range of application areas, including computer graphics, computational engineering, and medical imaging. Both unoptimised and optimised meshes for each test case are shown in Figures 6.14 and 6.20 in which elements are coloured according to shape quality. Specifically, elements with dihedral-angles satisfing $\theta(\tau) < 10.00^\circ$ and $\theta(\tau) > 160.0^\circ$ are shaded red, those satisfying $10.00^\circ \leq \theta(\tau) < 20.00^\circ$ or $140.0^\circ \leq \theta(\tau) < 160.0^\circ$ are shaded yellow, and those satisfying $20.00^\circ \leq \theta(\tau) < 30.00^\circ$ or $120.0^\circ \leq \theta(\tau) < 140.0^\circ$ are shaded green. Gray elements are of better quality. Additionally, histograms of element volume-length $v(\tau)$ and dihedral angle $\theta(\tau)$ measures are shown in each case, indicating mean and minimum volume-length ratios $\overline{v(\tau)}$, $v(\tau)_{\min}$, and minimum and maximum dihedral angle bounds $\theta(\tau)_{\min}$, $\theta(\tau)_{\max}$. Due to the difficulty associated with the optimisation of tetrahedral tessellations, the new mesh improvement strategy developed in this thesis was compared with the existing `STELLAR` optimisation package of Klinger and Shewchuk [19, 20]. Equivalent results generated using the `STELLAR` package are included in Figures 6.14 and 6.20. Note also that twelve of the benchmark problems investigated here are taken from the `STELLAR` verification set directly. The optimisation runs were completed using a vertex movement tolerance $\gamma = 1 \times 10^{-3}$ and a vertex smoothing threshold $\beta = 0.7$. The volumetric mesh improvement program was implemented in C++ and compiled as a 64-bit executable. The `STELLAR` program was set to use the element volume-length ratios as its objective function, and was run with default settings otherwise.

Analysis of Figures 6.14 and 6.20 show that the new volumetric mesh improvement program is effective in practice, significantly improving the quality of all meshes in the test sets. The `STELLAR` package is also seen to produce similar results in the majority of cases, with a few important exceptions. Focusing firstly on the test cases included in Figure 6.14, generated using the Frontal-Delaunay meshing program developed in Chapter 5, it is seen that both optimisation frameworks are typically successful in (i) removing the multitude of low quality *sliver* elements present in the initial meshes, (ii) improving both the minimum and mean volume-length ratios $v(\tau)_{\min}$, $\overline{v(\tau)}$, and (iii) dramatically improving the distribution of element dihedral angles $\theta(\tau)$. Overall, the new optimisation framework compares favourably to the `STELLAR` package on this first

Figure 6.14: Volumetric optimisation results for the SPHERE and ELLIPSOID meshes. Columns show the unoptimised input (left), results due to the new `JITTERBUG` library developed in the present study (middle) and results due to the `STELLAR` program (right). Optimisation was performed using a convergence tolerance $\gamma = 1 \times 10^{-3}$ and a smoothing threshold $\beta = 0.7$. Elements are coloured according to the distribution of dihedral-angles $\theta(\tau)$, with elements satisfying $\theta(\tau) < 10.00^\circ$ and $\theta(\tau) > 160.0^\circ$ shaded red, those satisfying $10.00^\circ \leq \theta(\tau) < 20.00^\circ$ or $140.0^\circ \leq \theta(\tau) < 160.0^\circ$ shaded yellow, and those satisfying $20.00^\circ \leq \theta(\tau) < 30.00^\circ$ or $120.0^\circ \leq \theta(\tau) < 140.0^\circ$ shaded green. Gray elements are of better quality. Normalised histograms of element volume-length $v(\tau)$ and element dihedral angle $\theta(\tau)$ are also included, in addition to statistics for the total runtime $t(s)$ and required outer-iterations for both optimisation programs.

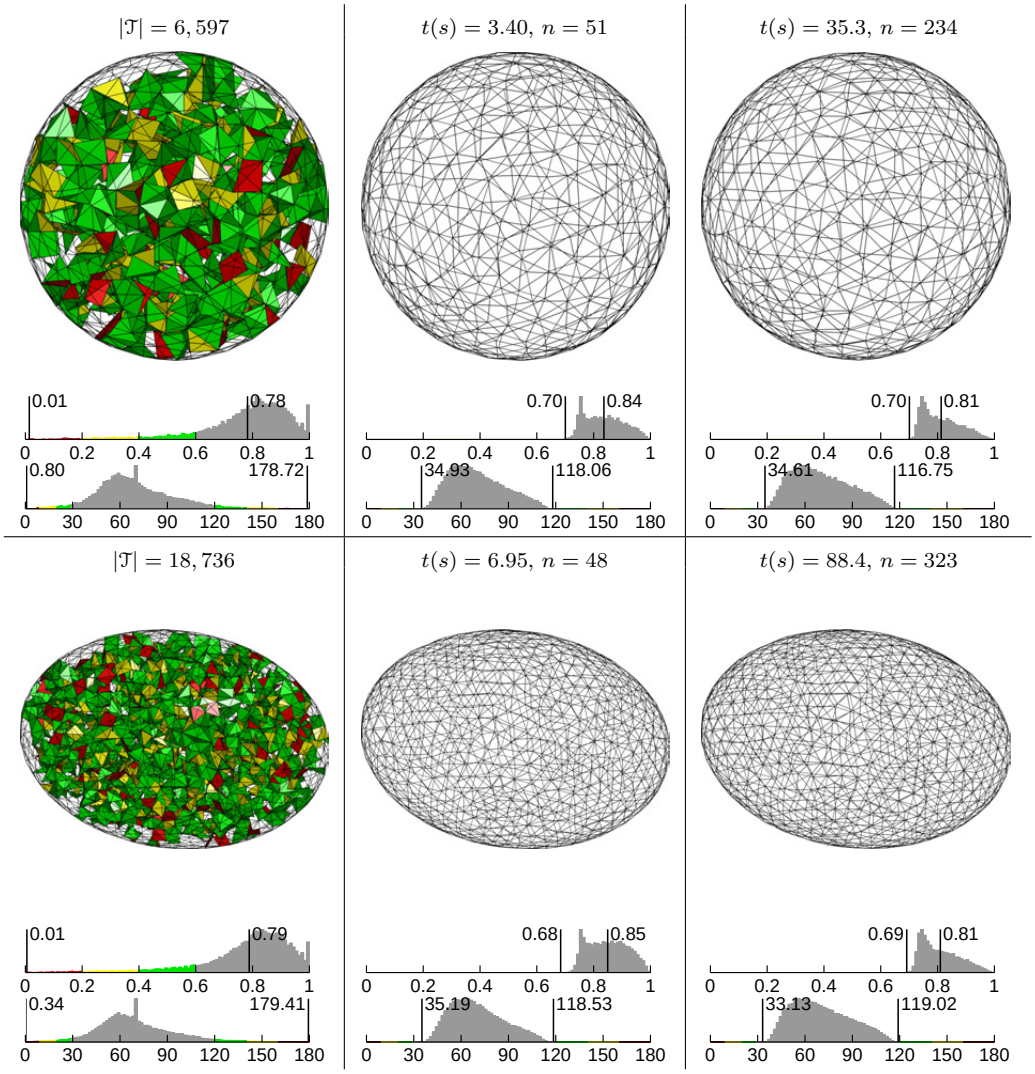


Figure 6.15: Continuation of Figure 6.14, showing comparative results for the FEMUR, HIP and VERTEBRA test problems.

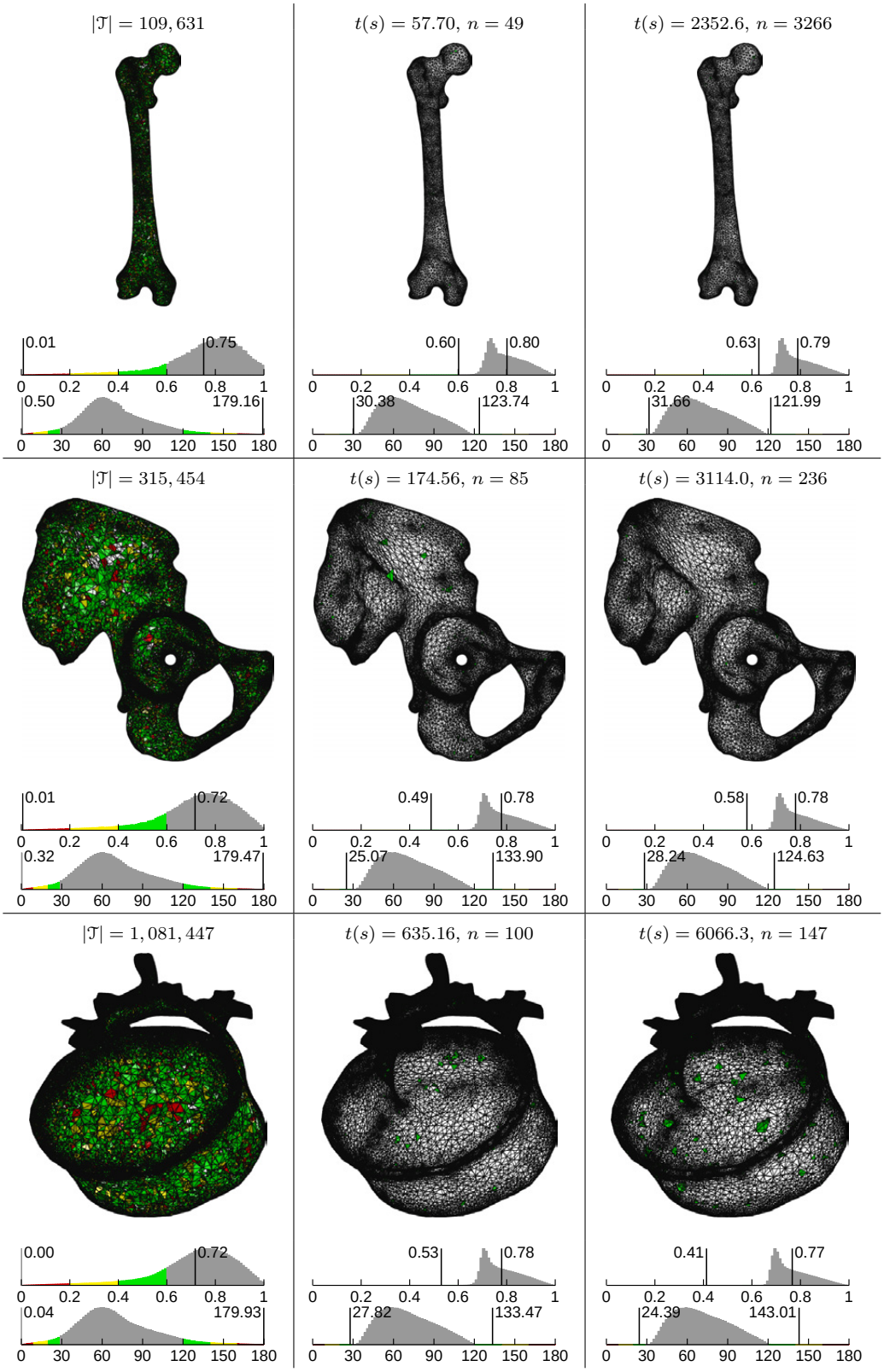


Figure 6.16: Continuation of Figure 6.14, showing comparative results for the BLADE, ROCKER and WOODTHINKER test problems.

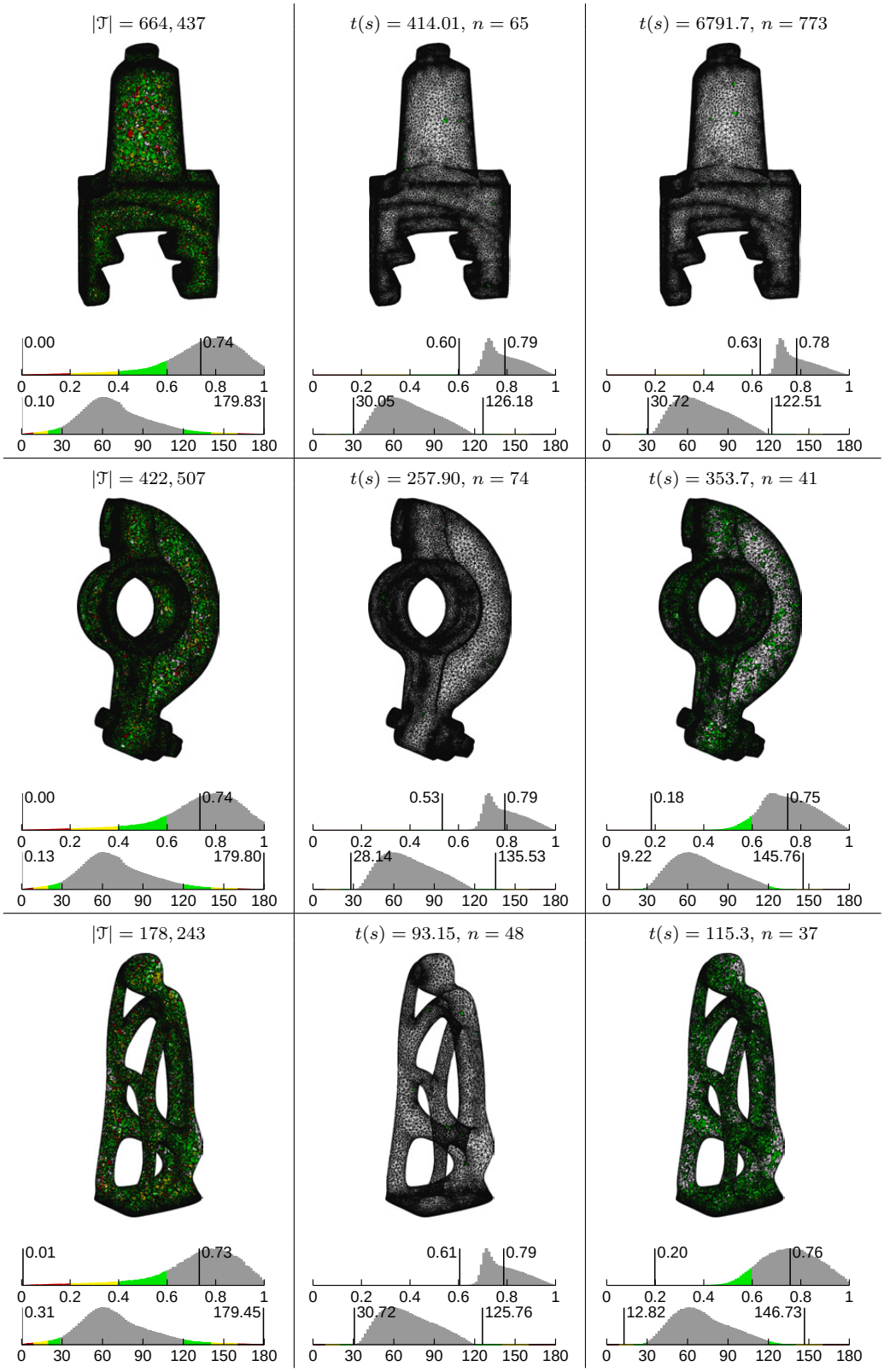


Figure 6.17: Continuation of Figure 6.14, showing comparative results for the HAND, SPIRAL 1 and SPIRAL 2 test problems.

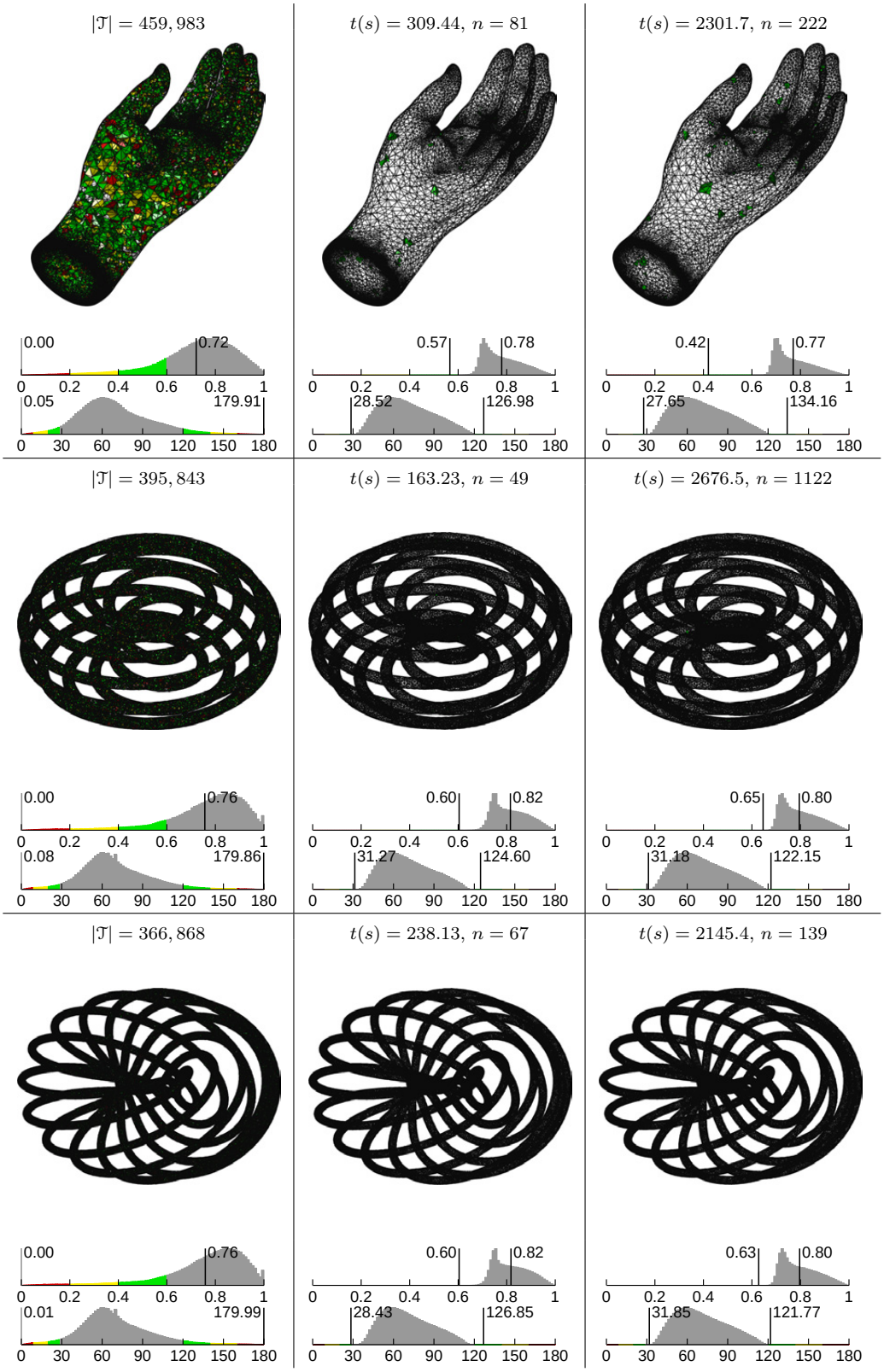


Figure 6.18: Continuation of Figure 6.14, showing comparative results for the BUNNY, BIMBA and VENUS test problems.

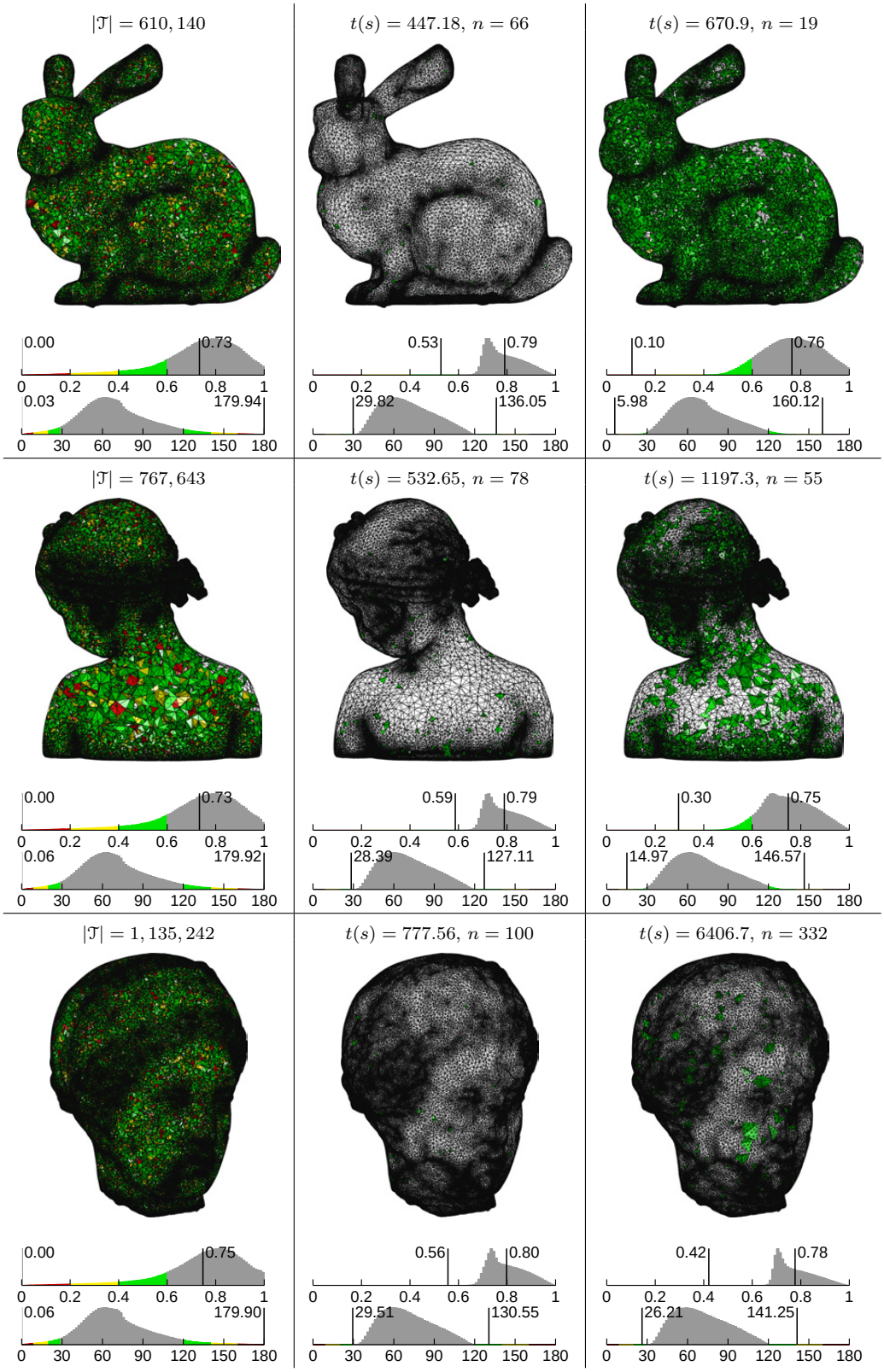
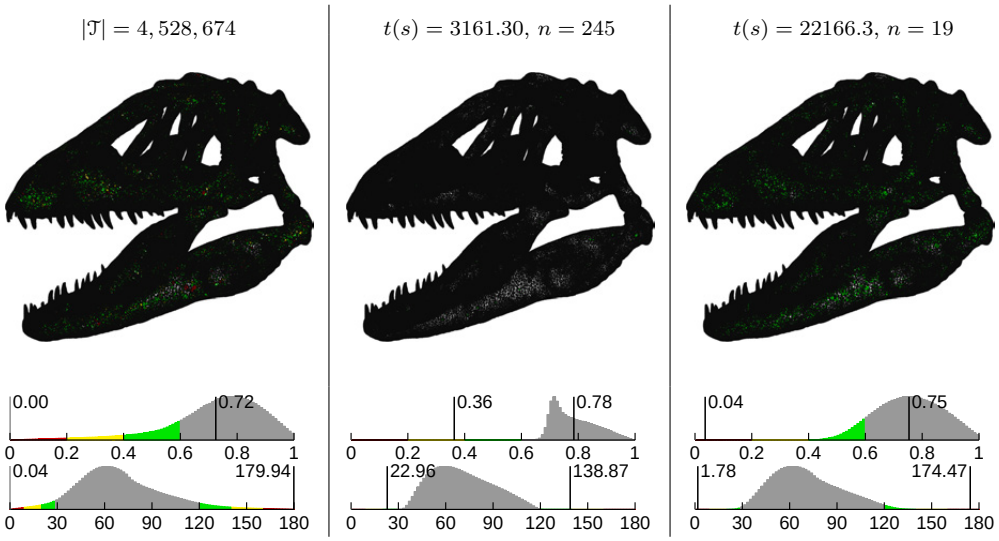


Figure 6.19: Continuation of Figure 6.14, showing comparative results for the DINOSAUR test problem.



set of benchmark problems in terms of optimisation quality, with the STELLAR program producing sub-optimal output for the ROCKER, WOODTHINKER, BUNNY, BIMBA and DINOSAUR problems. It is unclear why such a degradation in the performance of STELLAR is observed for these test cases, although it is noted that STELLAR often appears to exit early for these problems, performing significantly fewer iterations compared to other test-cases. For all other problems, the performance of the two packages is comparable, with worst case element dihedral angles improved to the range $20° \leq \theta(\tau) \leq 140°$. Consistent with the original findings of Klinger and Shewchuk [20], it is seen that both optimisation programs achieve best case results in which the dihedral angle distribution is restricted to the range $30° \leq \theta(\tau) \leq 120°$, or better.

Analysis of the test-cases included in Figure 6.20 – provided as verification examples for the STELLAR package – leads to slightly different conclusions. Consistent with previous results, it is observed that both optimisation packages are successful in improving both the volume-length $v(\tau)$ and dihedral angle $\theta(\tau)$ distributions for all problems, ensuring that low-quality sliver elements are removed from the optimised output. A more detailed analysis shows that the relative performance of the two optimisation packages is reversed within this second set of benchmark problems, with the STELLAR program producing better results for a range of individual test cases. While the optimisation framework proposed in this thesis is still reasonably successful across the full set of test cases, it does not always produce output in which the extreme dihedral angle values $\theta(\tau)_{\min}$ and $\theta(\tau)_{\max}$ are improved to the same level as that achieved by the STELLAR program. Specifically, it is noted that significant performance discrepancies exist when the quality of the initial tessellation is *very low*, as illustrated by the results for the TIRE, TFIRE, RAND1, RAND2, P and HOUSE2 problems. It is expected that the STELLAR package benefits from its sophisticated set of vertex insertion-based optimisation predicates in these cases.

Figure 6.20: Volumetric optimisation results for test cases introduced by Klinger and Shewchuk, showing the CUBE1K and CUBE10K meshes. Columns show the unoptimised input (left), results due to the new `JITTERBUG` library developed in the present study (middle) and results due to the `STELLAR` program (right). Optimisation was performed using a convergence tolerance $\gamma = 1 \times 10^{-3}$ and a smoothing threshold $\beta = 0.7$. Elements are coloured according to the distribution of dihedral-angles $\theta(\tau)$, with elements satisfying $\theta(\tau) < 10.00^\circ$ and $\theta(\tau) > 160.0^\circ$ shaded red, those satisfying $10.00^\circ \leq \theta(\tau) < 20.00^\circ$ or $140.0^\circ \leq \theta(\tau) < 160.0^\circ$ shaded yellow, and those satisfying $20.00^\circ \leq \theta(\tau) < 30.00^\circ$ or $120.0^\circ \leq \theta(\tau) < 140.0^\circ$ shaded green. Gray elements are of better quality. Normalised histograms of element volume-length $v(\tau)$ and element dihedral angle $\theta(\tau)$ are also included, in addition to statistics for the total runtime $t(s)$ and required outer-iterations for both optimisation programs.

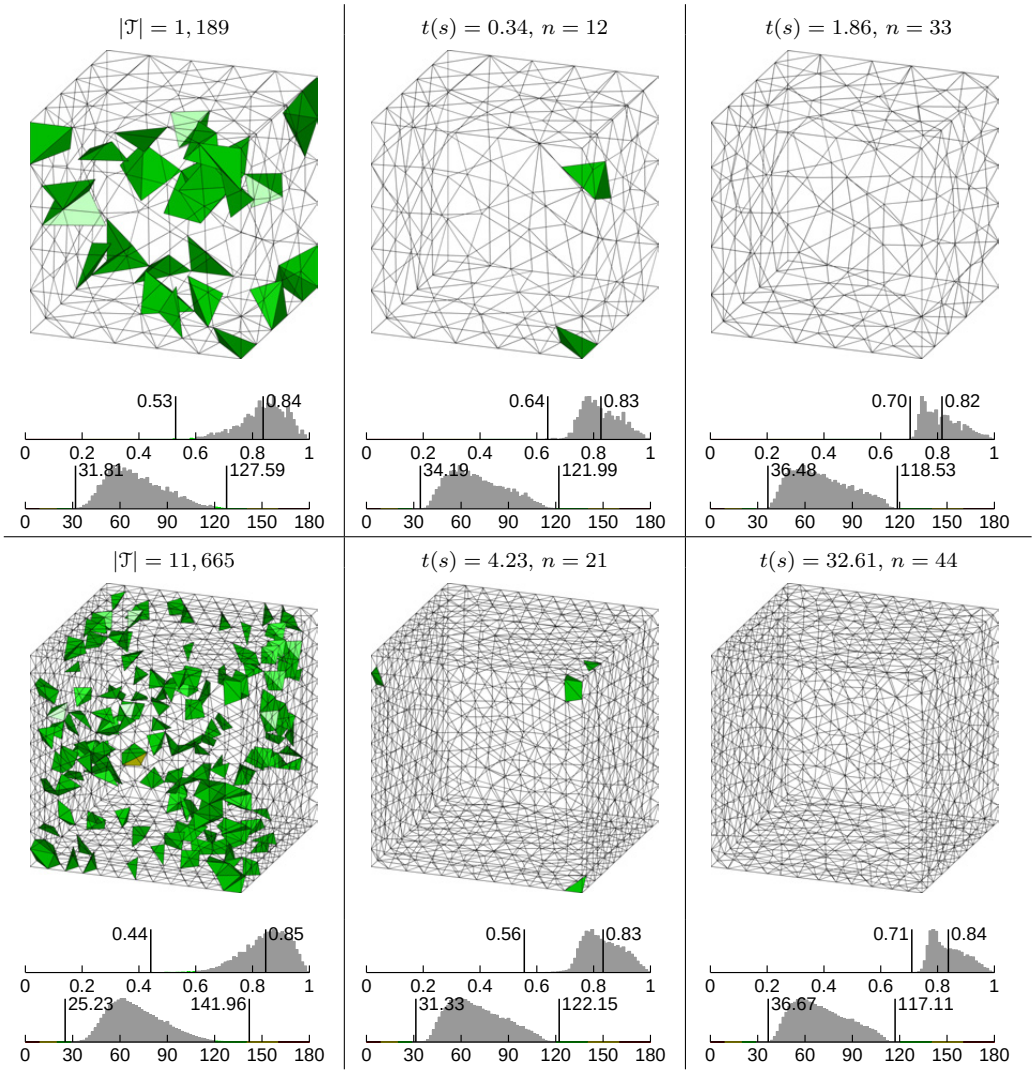


Figure 6.21: Continuation of Figure 6.20, showing comparative results for the TFIRE, TIRE and RAND 1 test problems.

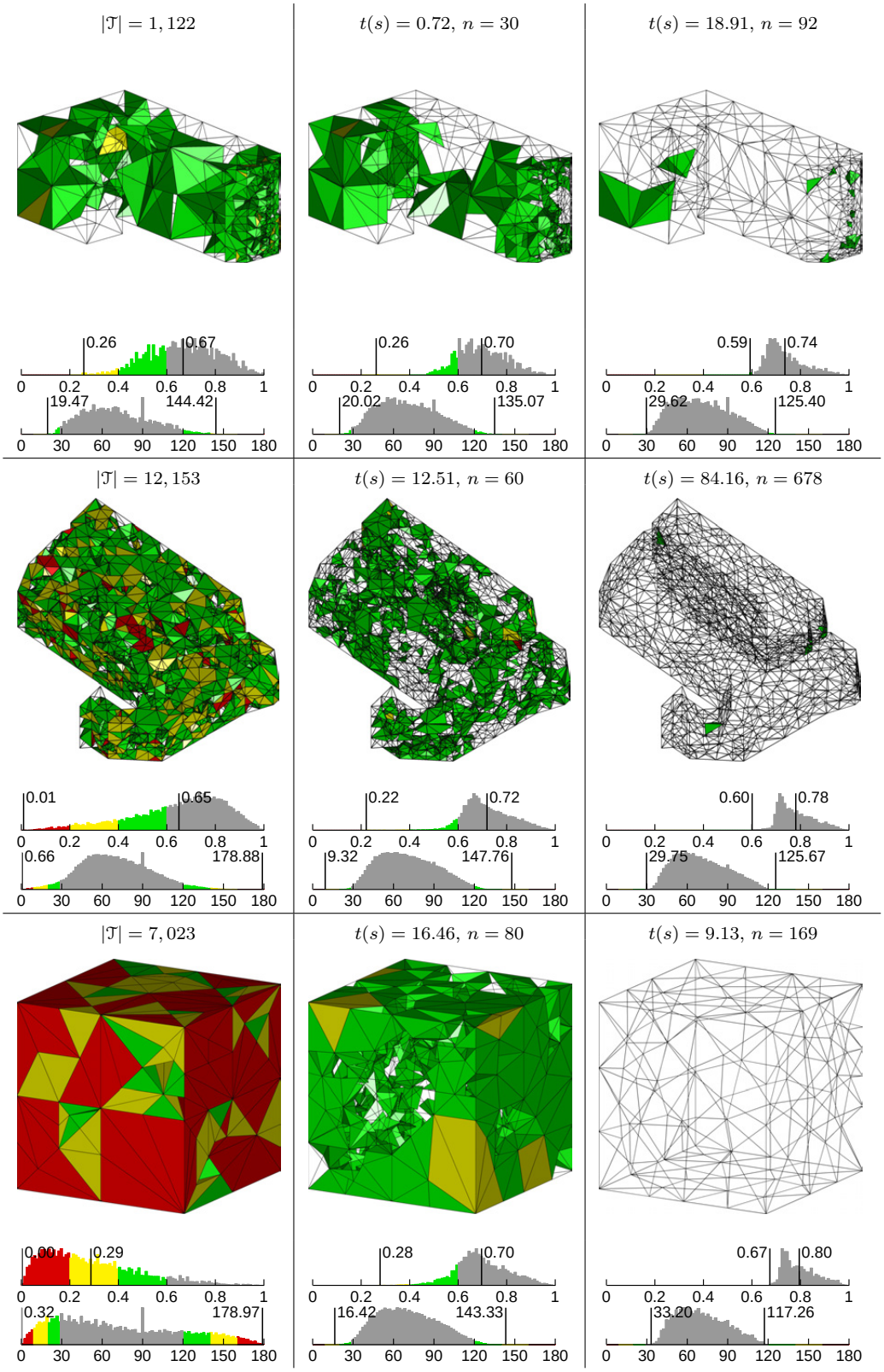


Figure 6.22: Continuation of Figure 6.20, showing comparative results for the RAND 2, DRAGON and COW test problems.

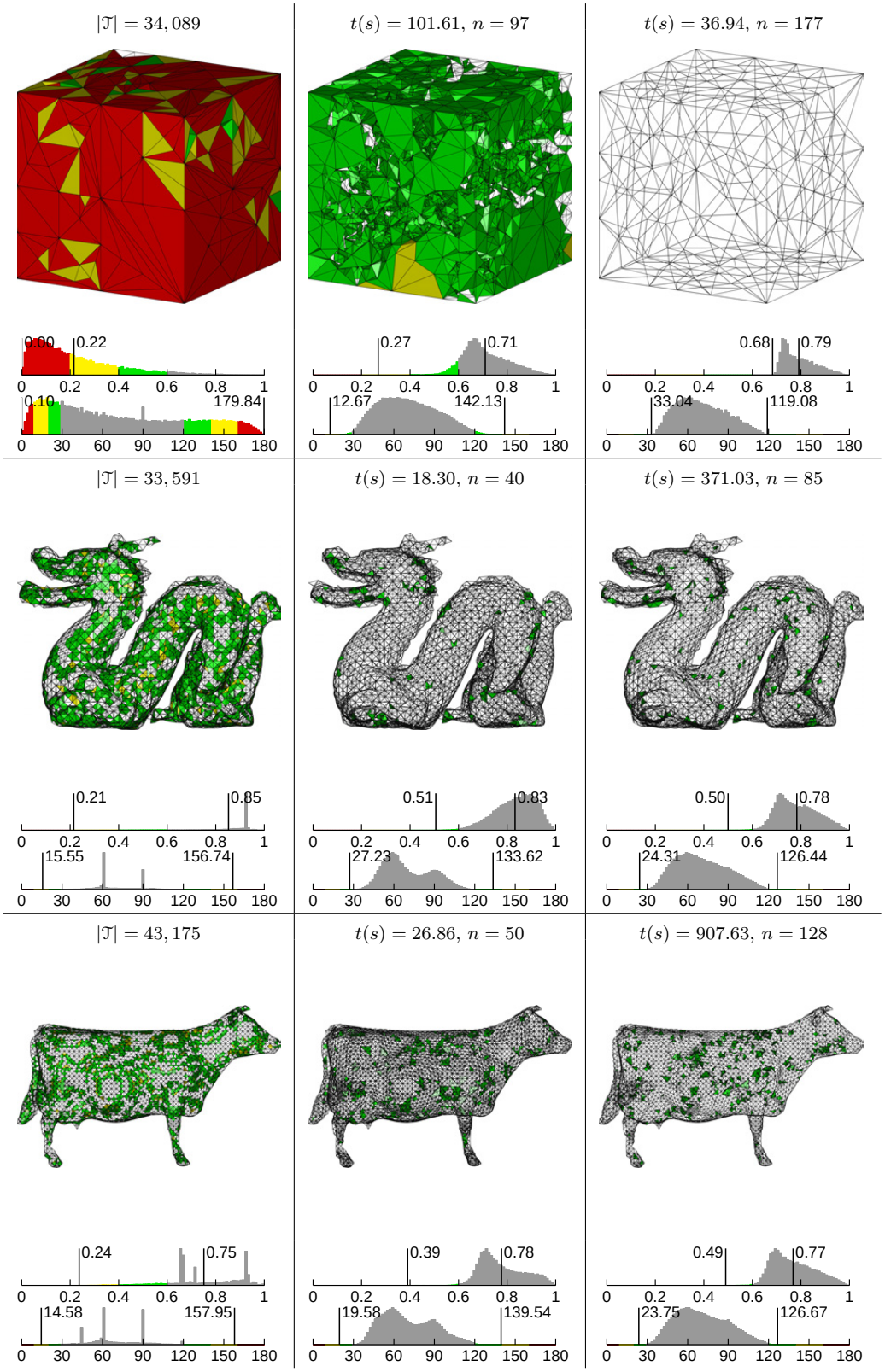


Figure 6.23: Continuation of Figure 6.20, showing comparative results for the P, St gallen and house test problems.

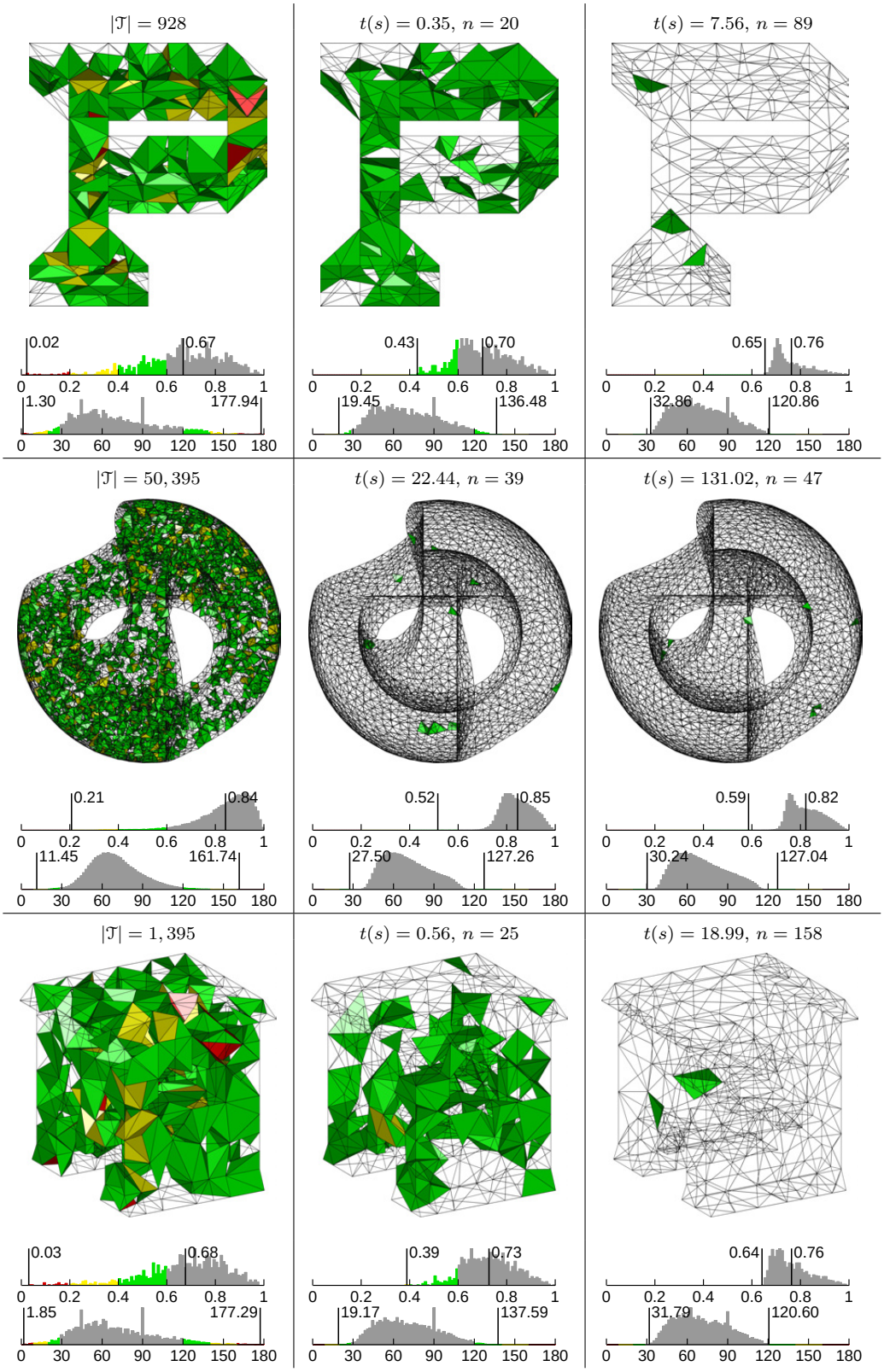
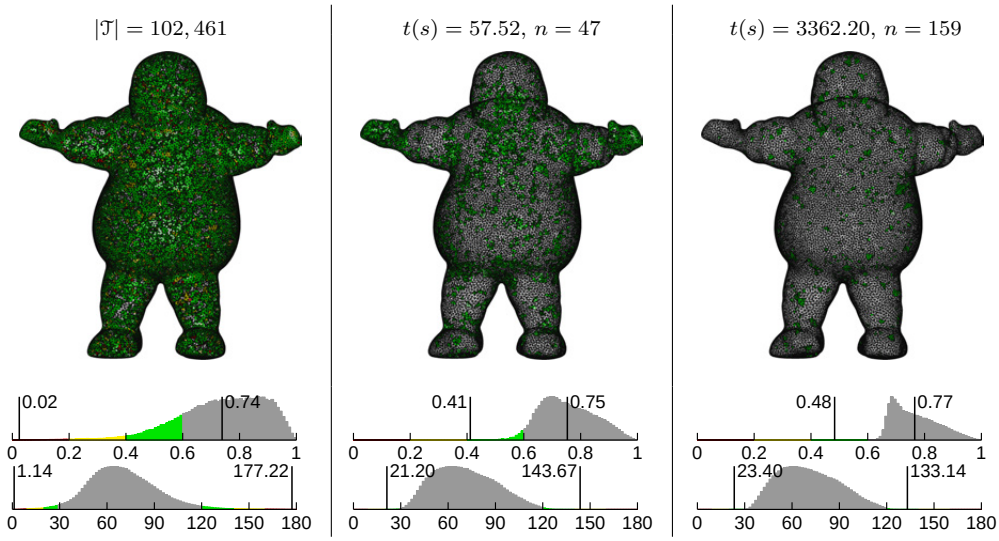


Figure 6.24: Continuation of Figure 6.20, showing comparative results for the STAYPUFT test problems.



In addition to optimisation quality, an analysis of the program run-times and iteration counts reveal significant differences in performance. Across all test problems, except for the RAND benchmarks, it can be seen that the `STELLAR` program is significantly slower than the new method proposed in this thesis, with the total runtime of the `STELLAR` package often as much as an order of magnitude larger. The `STELLAR` package was fastest only for the RAND test cases, where it produced significantly smaller tessellations. Noting that the run-times associated with `STELLAR` are typically large – measuring several hours for the large test problems studied – an order of magnitude decrease in computational cost represents an important improvement in the practicality of large-scale mesh optimisation. In addition to run-time metrics, it can be seen that the convergence of `STELLAR` appears to be somewhat erratic, with large differences in the required number of iterations-to-convergence observed between test cases.

6.6.1 Impact of Optimisation Features

Due to the difficulties associated with tetrahedral mesh optimisation, a detailed study of the effectiveness of the various mesh optimisation predicates is presented in Figures 6.25 and 6.27 for the BUNNY and VENUS test cases.

6.6.1.1 Optimisation Predicates

In Figure 6.25, the effect of the various topological-, smoothing- and refinement-based predicates is assessed, with both test-cases run using a combination of active and in-active optimisation predicates. Overall, similar behaviour was recorded for both the BUNNY and VENUS test-cases. Firstly, it can be seen that the application of topological transformation alone was effective in removing low-quality sliver elements from the tessellations, with the worst-case dihedral angles improved such that $15^\circ \leq \theta(\tau) \leq 155^\circ$ for both test-cases. Noting that the initial mesh topology was fully Delaunay in both cases, these results

Figure 6.25: Volumetric optimisation feature study for the BUNNY mesh, illustrating the effect of various combinations of vertex smoothing, topological transformation and vertex insertion operations. Elements are coloured according to shape-quality, consistent with previous figures. Normalised histograms of element volume-length $v(\tau)$ (left) and dihedral angle $\theta(\tau)$ (right) metrics are also shown.

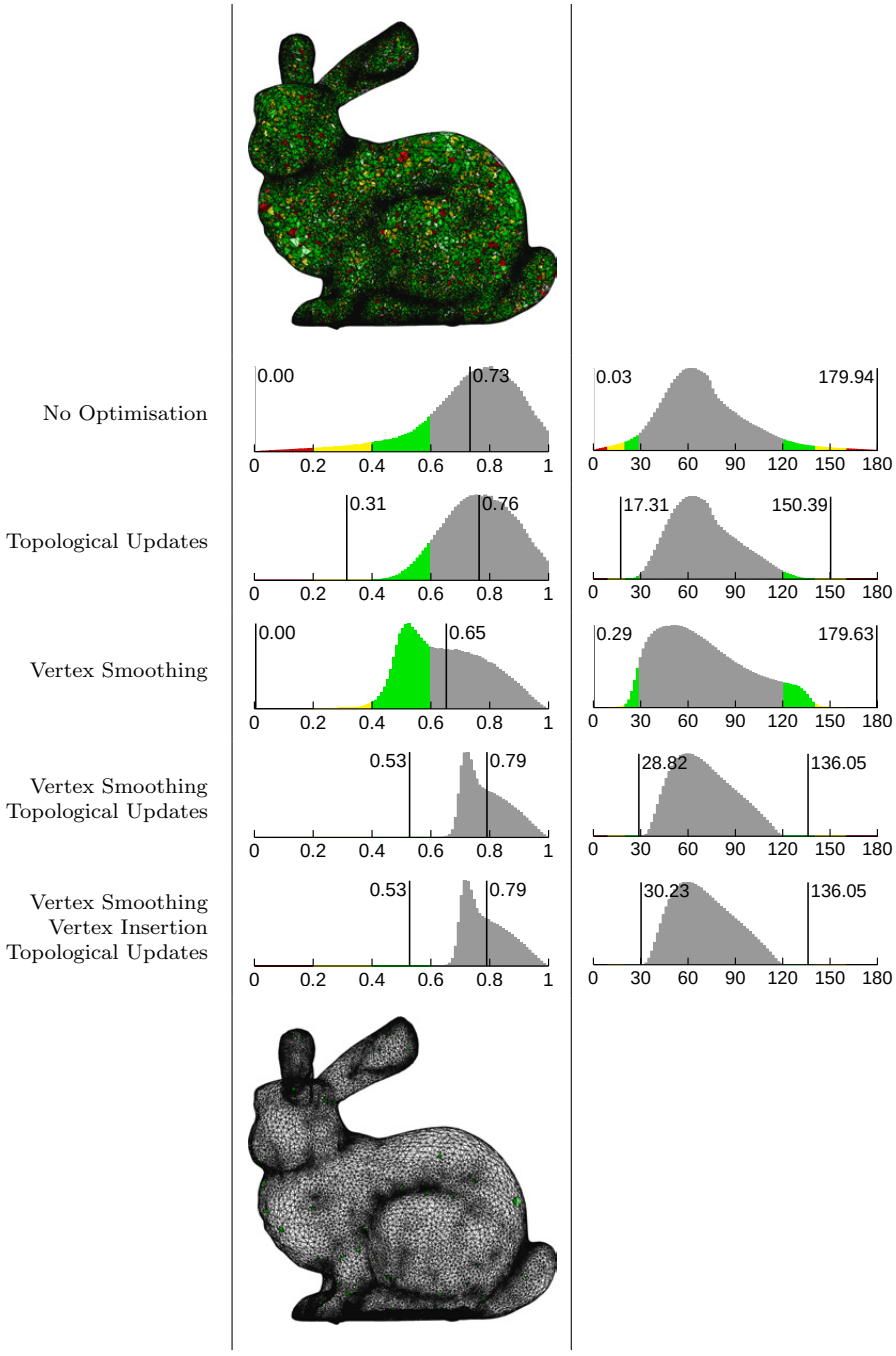
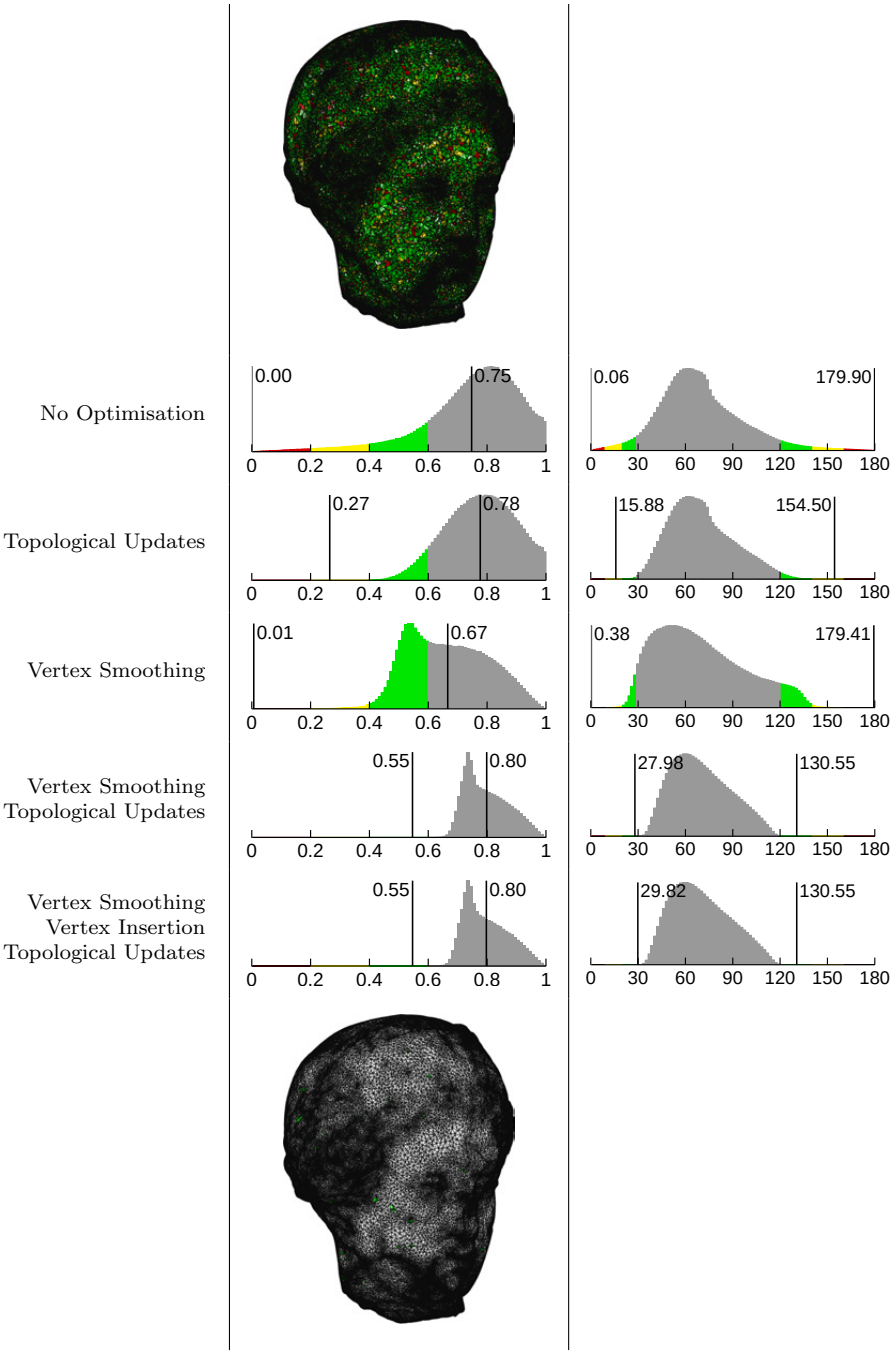


Figure 6.26: Volumetric optimisation feature study for the VENUS mesh, illustrating the effect of various combinations of vertex smoothing, topological transformation and vertex insertion operations. Elements are coloured according to shape-quality, consistent with previous figures. Normalised histograms of element volume-length $v(\tau)$ (left) and dihedral angle $\theta(\tau)$ (right) metrics are also shown.



reinforce the importance of seeking optimal, non-Delaunay topologies for tetrahedral meshing problems. Based on the success of topology-only optimisation methods, the development of enhanced topological transformations is clearly an interesting avenue for future investigation.

Conversely, the application of vertex-smoothing operations alone is shown to preserve many low-quality sliver elements, though it can also be seen that an improvement in the shape of the angle distributions is achieved. The combination of vertex-smoothing and topological optimisation clearly leads to a significant improvement in all quality metrics, easily outperforming either of the topology-only or smoothing-only configurations. Finally, the introduction of edge-refinement operations is shown to slightly improve mesh quality. It is expected that the use of more sophisticated vertex-insertion schemes, such as those suggested by Klinger and Shewchuk [19, 20], may significantly improve the effectiveness of insertion-based optimisation.

6.6.1.2 Vertex Smoothing Threshold

In Figure 6.27, the effect of the combined vertex smoothing operation is assessed. Firstly, recall that the vertex optimisation predicate introduced in Section 6.2.3 is a hybrid procedure, using a simple Laplacian-smoothing update when the local mesh quality is sufficiently high and an update based on local non-smooth optimisation otherwise. Specifically, the vertex-centred quality metric $\mathcal{Q}_{\mathbf{x}_i} = \min(\mathcal{Q}_{t_j}(\mathbf{x}))$ is required to exceed a user-specified threshold β to invoke a Laplacian-smoothing-based update. Considering again the BUNNY and VENUS benchmark problems in detail, the effect of varying the threshold β over the range $[1/2, 1]$ is investigated in Figure 6.27. Other than the variation in β, the full optimisation schedule, incorporating the various topological transformation, vertex smoothing and edge-refinement predicates is used. Results for both benchmark problems show similar trends. An analysis of Figure 6.27 shows that firstly, as expected, mesh quality is typically improved with increasing β, with the proportion of vertex updates due to the optimal non-smooth optimisation process increased accordingly. Despite these improvements, it is also clear that beyond $\beta \simeq 0.7$ there is little improvement in either the minimum volume length metrics or the worst-case dihedral angles. Importantly, it is also noted that for $\beta \geq 0.7$ there is a corresponding *reduction* in the mean volume-length metrics. For both test-cases, it can be seen that $\overline{v(\tau)}$ approaches $\simeq 0.8$ as $\beta \to 1$.

These results demonstrate that, contrary to planar and surface triangulations, it is typically very difficult to construct high quality tetrahedral tilings for general volumetric domains. Eppstein, Sullivan, and Üngör [8] have shown that even for simple *slab*-like configurations it is difficult to construct tessellations for which all elements exhibit both good angle distributions and good size regularity. As a result, it is expected that for higher values of β there is a reduction in $\overline{v(\tau)}$ as high-quality elements are 'sacrificed' to improve the minimum quality elements. Lastly, recalling that a *worst-first* vertex metric $\mathcal{Q}_{\mathbf{x}_i} = \min(\mathcal{Q}_{t_j}(\mathbf{x}))$ is used in this work, I note that alternate 'mean' metrics, such as the Centroidal-Voronoi [1, 7] or Optimal-Delaunay [3, 4, 5] formulations, may lead to improvements in mean mesh quality in some cases [5]. Note that such improvements in the mean measures are often achieved at the expense of the worst-case elements in the

Figure 6.27: Volumetric optimisation threshold study for the BUNNY mesh, illustrating the effect of the vertex smoothing threshold, β, on mesh quality. Elements are coloured according to shape-quality, consistent with previous figures. Normalised histograms of element volume-length $v(\tau)$ (left) and dihedral angle $\theta(\tau)$ (right) metrics are also shown.

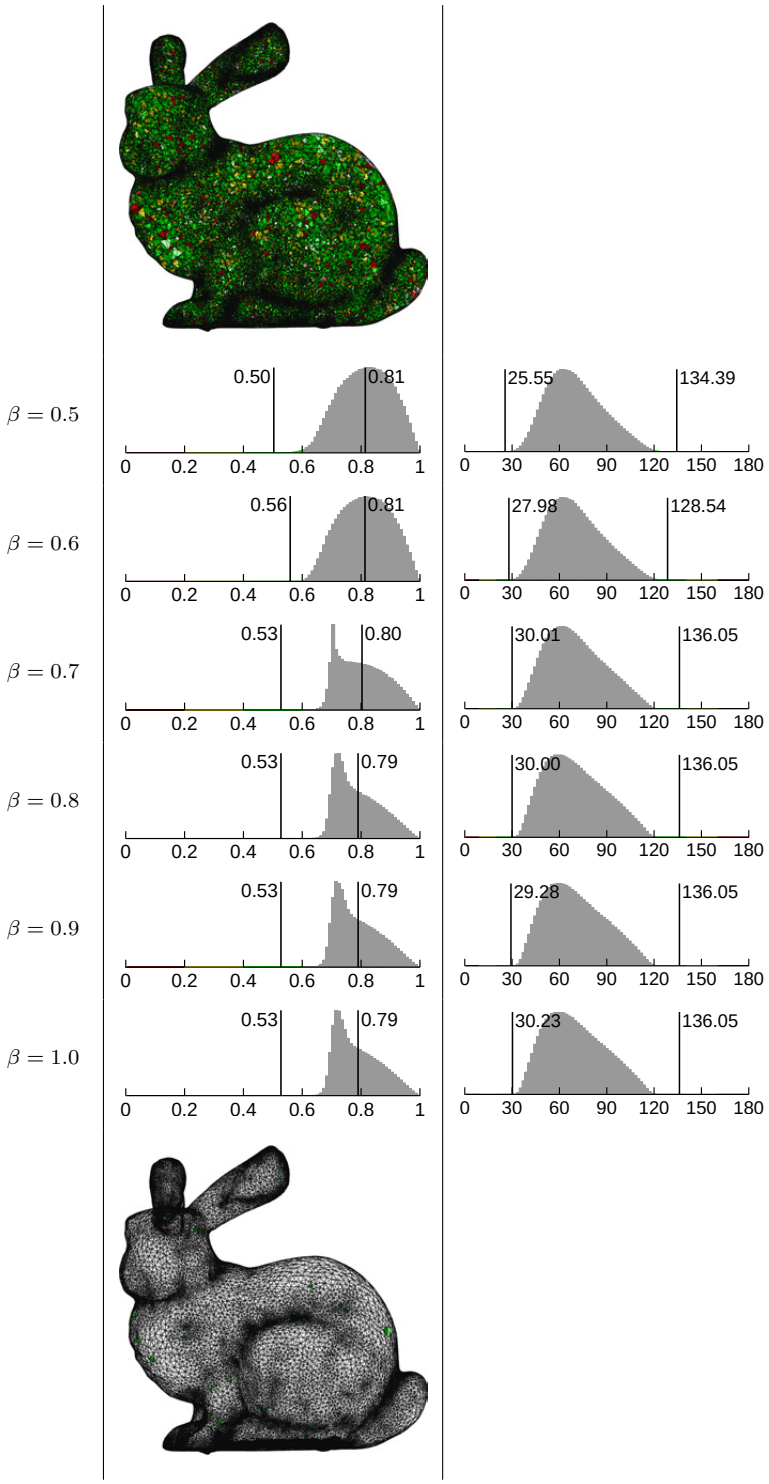
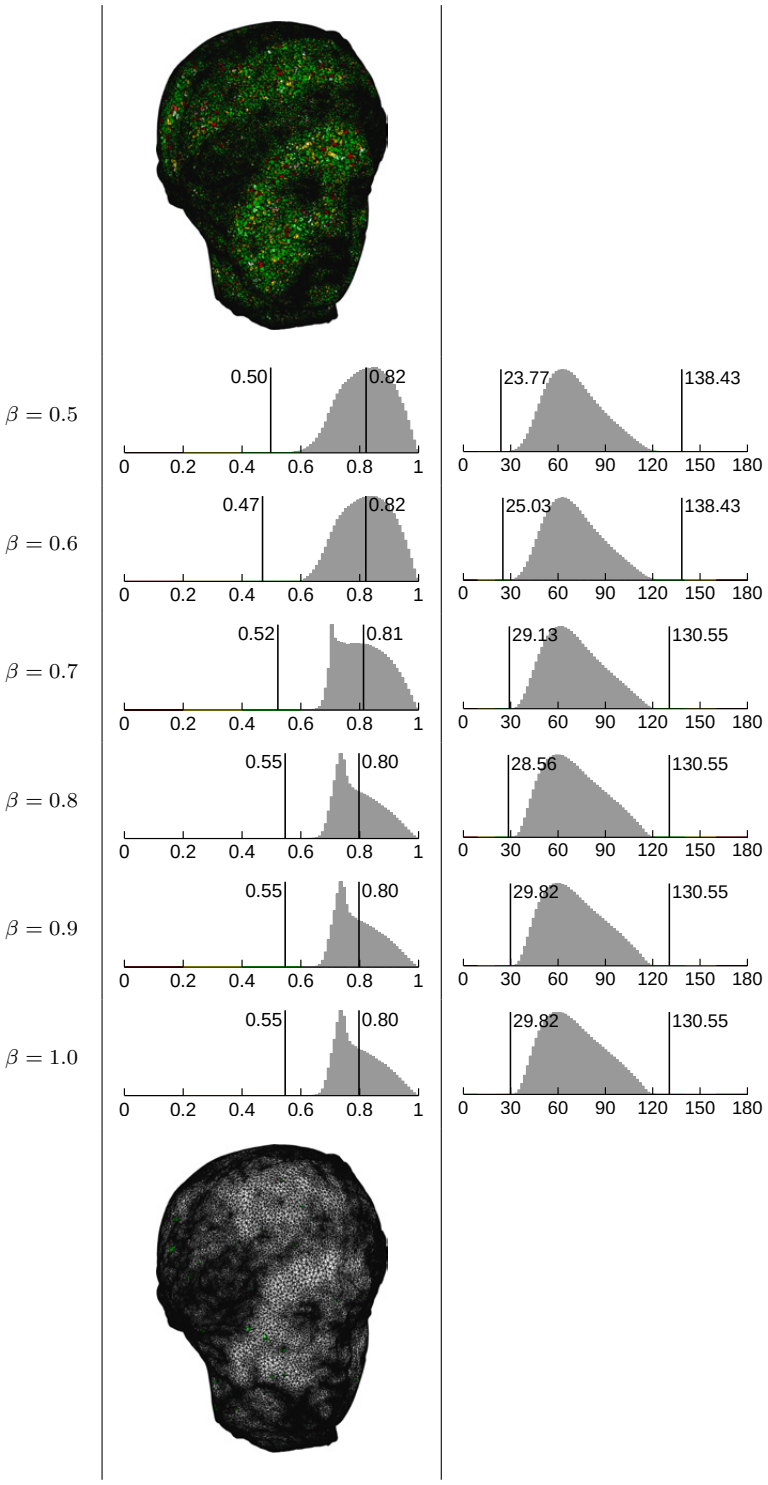


Figure 6.28: Volumetric optimisation threshold study for the VENUS mesh, illustrating the effect of the vertex smoothing threshold, β, on mesh quality. Elements are coloured according to shape-quality, consistent with previous figures. Normalised histograms of element volume-length $v(\tau)$ (left) and dihedral angle $\theta(\tau)$ (right) metrics are also shown.



tessellation.

6.7 Conclusions

In this chapter, I have presented a new mesh improvement framework for the optimisation of planar, surface and volumetric simplicial tessellations. The mesh improvement program is based on a local 'hill-climbing' optimisation paradigm, and relies on a series of geometric, topological and refinement-based operations. Building on the previous work of Freitag and Olivier-Gooch [13] and Klinger and Shewchuk [20], I have presented a detailed description of the various optimisation predicates and have outlined the manner in which they are interleaved to form a full optimisation schedule. I have proposed a new priority-driven strategy, in which optimisation is targeted at the worst entities in the mesh at each iteration. Specifically, I have shown that by restricting the various optimisation predicates to neighbourhoods adjacent to an *active-set* of vertices, the new optimisation program can achieve a significant reduction in total computational effort while maintaining very high levels of optimisation quality. A series of comparative experimental studies have confirmed the effectiveness of the new mesh improvement program in practice, with high-quality results achieved for a variety of planar, surface and volumetric test cases. Due to the difficulties associated with the improvement of tetrahedral meshes, a set of detailed comparisons were made with the `STELLAR` optimisation program of Klinger and Shewchuk [19, 20]. A comparative analysis showed, firstly, that for relatively high-quality input, both optimisation programs typically generated similar, and very high-quality output. When applied to low-quality input, the `STELLAR` package was found to often outperform the proposed method. Across all benchmark problems the proposed optimisation program was found be faster than the existing package, often outperforming `STELLAR` by an order of magnitude or more. Results show that, given output from the Frontal-Delaunay or Delaunay-refinement meshing algorithms presented in previous chapters, the new mesh optimisation program generates very high-quality output at low computational expense.

A number of avenues for future investigation have also been identified, including: (i) the search for enhanced topological transformations for tetrahedral complexes, (ii) the development of improved refinement-based insertion predicates, following the formulation presented by Klinger and Shewchuk [20], and (iii) the investigation of alternate vertex-centred quality metrics for smoothing operations, designed to improve mean mesh quality. Significant opportunities also exist to explore the application of the existing optimisation strategy to non-simplicial mesh types, such as mixed quadrilateral- and hexahedral-based tessellations.

References

[1] Alliez, P., Cohen-Steiner, D., Yvinec, M., and Desbrun, M. Variational Tetrahedral Meshing. *ACM Transactions on Graphics 24*, 3 (2005), 617–625.

[2] Briére de L'isle, E., and George, P. L. Optimization of tetrahedral meshes. In *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, I. Babuska, W. D. Henshaw, J. E. Oliger, J. E. Flaherty, J. E. Hopcroft, and T. Tezduyar, Eds., vol. 75 of *The IMA Volumes in Mathematics and its Applications*. Springer New York, 1995, pp. 97–127.

[3] Chen, L., and Holst, M. Efficient mesh optimization schemes based on optimal Delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering 200*, 9 (2011), 967–984.

[4] Chen, L., and Xu, J. Optimal Delaunay Triangulations. *Journal of Computational Mathematics 22*, 2 (2004).

[5] Chen, Z., Wang, W., Lévy, B., Liu, L., and Sun, F. Revisiting Optimal Delaunay Triangulation for 3D Graded Mesh Generation. *SIAM Journal on Scientific Computing 36*, 3 (2014), A930–A954.

[6] de Cougny, H. L., and Shephard, M. S. Refinement, Derefinement, and Optimization of Tetrahedral Geometric Triangulations in Three Dimensions. 1995.

[7] Du, Q., Faber, V., and Gunzburger, M. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review 41*, 4 (1999), 637–676.

[8] Eppstein, D., Sullivan, J. M., and Üngör, A. Tiling space and slabs with acute tetrahedra. *Computational Geometry 27*, 3 (2004), 237–255.

[9] Field, D. A. Laplacian Smoothing and Delaunay Triangulations. *Communications in applied numerical methods 4*, 6 (1988), 709–712.

[10] Freitag, L., Jones, M., and Plassmann, P. A Parallel Algorithm for Mesh Smoothing. *SIAM Journal on Scientific Computing 20*, 6 (1999), 2023–2040.

[11] Freitag, L. A. On combining Laplacian and optimization-based mesh smoothing techniques. *ASME Applied Mechanics Division-Publications-AMD 220* (1997), 37–44.

[12] Freitag, L. A., and Knupp, P. M. Tetrahedral Mesh Improvement via Optimization of the Element Condition Number. *International Journal for Numerical Methods in Engineering 53*, 6 (2002), 1377–1391.

[13] Freitag, L. A., and Ollivier-Gooch, C. Tetrahedral Mesh Improvement using Swapping and Smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[14] Frey, P. J., and Alauzet, F. Anisotropic Mesh Adaptation for CFD Computations. *Computer methods in applied mechanics and engineering 194*, 48 (2005), 5068–5082.

[15] Hurtado, F., and Noy, M. Graph of Triangulations of a Convex Polygon and Tree of Triangulations. *Computational Geometry 13*, 3 (1999), 179–188.

[16] Joe, B. Three-dimensional Triangulations from Local Transformations. *SIAM Journal on Scientific and Statistical Computing 10*, 4 (1989), 718–741.

[17] Joe, B. Construction of Three-dimensional Improved-Quality Triangulations Using Local Transformations. *SIAM Journal on Scientific Computing 16*, 6 (1995), 1292–1307.

[18] Klincsek, G. Minimal Triangulations of Polygonal Domains. *Ann. Discrete Math 9* (1980), 121–123.

[19] Klingner, B. M. *Tetrahedral Mesh Improvement*. PhD thesis, Berkeley, California, 2008.

[20] Klingner, B. M., and Shewchuk, J. R. Aggressive Tetrahedral Mesh Improvement. In *Proceedings of the 16th International Meshing Roundtable*, M. L. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 3–23.

[21] Lawson, C. L. Properties of N-dimensional Triangulations. *Comput. Aided Geom. Des. 3*, 4 (Jan. 1987), 231–246.

[22] Misztal, M. K., Bærentzen, J. A., Anton, F., and Erleben, K. Tetrahedral Mesh Improvement Using Multi-face Retriangulation. In *Proceedings of the 18th International Meshing Roundtable*, B. W. Clark, Ed. Springer Berlin Heidelberg, 2009, pp. 539–555.

[23] Shewchuk, J. R. Two Discrete Optimization Algorithms for the Topological Improvement of Tetrahedral Meshes. Tech. rep., University of California, Berkeley, Department of Computer Science, 2002.

Chapter 7

Conclusions

In this thesis, I have developed a range of methods designed to generate high-quality unstructured simplicial tessellations for domains in $\mathbb{R}^2$ and $\mathbb{R}^3$, including techniques that target the various planar, surface and volumetric problems typically encountered in practical computational modelling and simulation tasks. Expanding on previous work focused on the development of meshing algorithms deriving from the Delaunay-refinement paradigm, I have introduced a new set of Frontal-Delaunay meshing techniques that typically outperform existing methods in terms of both practical performance and theoretical robustness. Additionally, I have developed a new, computationally efficient mesh improvement framework designed to optimise the quality of existing triangular and tetrahedral meshes. When used in concert, these new mesh generation and mesh improvement algorithms constitute a complete meshing solution – able to create high-quality unstructured tessellations suitable for subsequent numerical simulation and modelling studies. In addition to a full theoretical exposition of the various theorems and algorithms introduced, I provide high-quality implementations of the new mesh generation and optimisation schemes in the `JIGSAW` and `JITTERBUG` packages, respectively. An extensive series of experimental studies have been used to confirm both the robustness and effectiveness of these new techniques in practice, verifying that the new algorithms are typically at least competitive, and are often found to outperform various existing state-of-the-art meshing techniques.

7.1 Frontal-Delaunay Mesh Generation

I have developed and implemented a number of algorithms for the construction of simplicial meshes based on a new Frontal-Delaunay paradigm – a hybridisation of conventional Delaunay-refinement and advancing-front techniques. The key contribution in this work is a generalisation of the point-placement strategy used when refining elements in a conventional Delaunay-refinement algorithm. Rather than forcing new Steiner points to lie at the element circumcentres only, as per conventional techniques, I have shown that a careful positioning of new vertices along the adjacent *faces* of the associated Voronoi complex can lead to a significant improvement in mesh quality, while simultaneously

maintaining the theoretical robustness associated with conventional circumcentre-based strategies. Specifically, I have shown that a new class of *size-optimal* Voronoi-centred refinement methods typically lead to substantial improvements in mesh shape-quality and structure when element size constraints are imposed. I have also demonstrated that these new techniques maintain existing bounds on element radius-edge ratios and surface discretisation errors. For planar cases, this new strategy is similar to the methods introduced by Rebay [18] and Üngör [20], though I am not aware of any previous generalisations to either surface or volumetric meshing problems.

While the new Voronoi-centred refinement algorithms follow the structure of a conventional Delaunay-refinement strategy, the nature of the new point-placement scheme shares many similarities with advancing-front algorithms, with both methods incrementally introducing new vertices about an existing set of *frontal* faces. Like advancing-front techniques, but contrary to the behaviour of conventional Delaunay-refinement methods, I have shown that the quality of meshes generated using the new Voronoi-centred Frontal-Delaunay algorithms improves with decreasing mesh size. Such behaviour is an important consideration for a range of numerical modelling and simulation applications, such as some problems in computational fluid dynamics and structural analysis that require very high mesh quality in locally aligned regions such as boundary layers. It is hoped that the combination of high element quality and theoretical robustness offered by the new Frontal-Delaunay techniques are of benefit to such fields.

7.2 Mesh Optimisation

I have developed and implemented a new mesh improvement framework for the optimisation of existing simplicial tessellations. The new optimisation programs are an evolution of previous work by Freitag and Olivier-Gooch [6] and Klinger and Shewchuk [9], and are designed to improve isotropic measures of element *shape-quality* via a combination of local topological, geometrical and refinement-based optimisation predicates. The main contribution in this thesis is the use of a new priority-driven optimisation schedule, in which computational work is minimised by concentrating optimisation effort on a local subset of mesh entities that are amenable to further improvement. Specifically, I have shown that by tracking and updating a set of *active* vertices throughout the optimisation procedure, the application of local optimisation predicates can be restricted to a narrow neighbourhood adjacent to vertices that have not yet reached a converged state. This approach has been shown to achieve a significant reduction in total computational effort while maintaining very high levels of optimisation quality.

A series of comparative experimental studies have confirmed the effectiveness of the new mesh improvement framework in practice, with high-quality results achieved for a variety of planar, surface and volumetric test cases. Additionally, a set of detailed comparisons were made with the STELLAR package, a state-of-the-art optimisation program due to Klinger and Shewchuk [8, 9]. A comparative analysis showed that while the STELLAR program produced slightly better output for a series of pathological examples, the new mesh improvement framework offered competitive optimisation performance at

greatly reduced computational expense for the majority of test cases examined. The performance of the new optimisation programs were found to be excellent when applied to meshes generated using the new Frontal-Delaunay algorithms developed in this thesis. The generation of high-quality tetrahedral meshes with bounded dihedral angle distributions has been an ongoing issue for various types of three-dimensional numerical simulation and modelling studies, and it is hoped that the new tetrahedral mesh optimisation program will be especially useful in these areas.

7.3 Topics for Future Investigation

Mesh generation is a broad and evolving area of research. Considering the ever increasing sophistication of associated methods in numerical modelling and simulation, computer graphics, animation and data analysis, it is necessary to continually pursue innovative developments in order to maintain pace. The topics included below are extensions of the methods presented in this thesis to new or evolving trends in the computational modelling community:

- **Parallel mesh generation and smoothing.** While the Delaunay-refinement and Frontal-Delaunay techniques presented in this thesis achieve optimal $O(n \log(n))$ algorithmic performance, additional efficiency may be pursued through the development of *parallel* meshing techniques, in which the computational burden is distributed between multiple processing units. Specifically, a number of authors, including Pirzadeh and Zagaris [17] and Linardakis and Chrisochoides [12, 13] have recently shown that *domain-decomposition* type approaches can be used to construct efficient parallel algorithms. An experimental implementation, based on a balanced decomposition of a coarse and weighted Delaunay tessellation of the bounding geometry, has been found to produce encouraging levels of parallel speed-up in a multi-threaded environment.
- **Enhanced Voronoi-centred refinement schemes.** It may be possible to further generalise the Voronoi-based point-placement schemes developed in this thesis to incorporate additional flexibility and constraints. Specifically, in the context of planar mesh generation, Erten and Üngör [4] have recently shown that substantial improvements in element quality can be achieved by using a set of generalised *off-centre* Steiner points, positioned at non-adjacent locations on the underlying Voronoi diagram. Additionally, Chernikov and Chrisochoides [2] have recently expanded upon a scheme originally due to Chew [3], and have shown that the introduction of a distributed *picking-region* for Steiner vertices in the local neighbourhood of element circumcentres can be used to limit the creation of low-quality sliver elements in tetrahedral Delaunay-refinement schemes.
- **Locally optimal tetrahedral mesh generation.** Despite provably-good behaviour for planar and surface-based triangulations in $\mathbb{R}^2$ and $\mathbb{R}^3$, it is known that the topology of the Delaunay tessellation is sub-optimal for higher-dimensional problems, including the tetrahedral case in $\mathbb{R}^3$. Rather than accepting the low-quality *sliver* elements associated with tetrahedral Delaunay-refinement or Frontal-Delaunay schemes, it may

instead be possible to pursue the development of locally optimal non-Delaunay meshing techniques, based on the topological transformations introduced in Chapter 6 in the context of mesh optimisation. Methods based on such an approach may share similarities with the *local-reconnection* algorithms of Marcum and Weatherill [15]. While such methods may indeed achieve improved performance in practice, it is expected that such behaviour may come at the expense of theoretical robustness and provable guarantees.

- **Anisotropic mesh generation and smoothing.** The behaviour of physical phenomena is often highly anisotropic and it is therefore sensible to incorporate such anisotropy into the underlying numerical models and simulations used to study such behaviour. Such methods require the generation of anisotropic meshes, in which the local measures of mesh quality are deformed and stretched according to a local metric field. Both the mesh generation and mesh improvement methods developed in this thesis could be extended to support such tessellations through the development of anisotropic mesh quality kernels. The development of such methods are ongoing, including contributions from Labelle and Shewchuk [10], Frey and Alauzet [7], Boissonnat, Shi, Tournois and Yvinec, [1] and Lévy and Bonneel [11].
- **High-order mesh generation and smoothing.** Many modern numerical schemes for computational modelling and simulation are formulated using high-order polynomial basis functions. In order to achieve their full rate of convergence, such schemes require the generation of high-order tessellations, consisting of elements with *curved* polynomial edges and faces. The mesh optimisation techniques developed in this thesis could be extended to support such tessellations through the development of high-order element quality metrics and optimisation predicates. Work in this area is ongoing, including contributions from Lu, Shephard, Tendulkar and Beall [14], Roca, Gargallo-Peiró and Sarrate [19] and Persson and Peraire [16].
- **Higher-dimensional mesh generation and smoothing.** The direct generation of unstructured meshes in higher dimensional spaces is beginning to receive significant attention, with recent studies describing techniques for the direct manipulation of unstructured four-dimensional space-time meshes for medical imaging [5], and unstructured Discontinuous-Galerkin finite-element methods [21]. The Delaunay-based mesh generation methods described in this thesis generalise directly to higher-dimensional spaces, though the so-called 'curse-of-dimensionality' is expected to substantially increase the associated level of computational work required. The development of higher-dimensional mesh optimisation predicates is expected to be a challenging task.

References

[1] Boissonnat, J. D., Shi, K. L., Tournois, J., and Yvinec, M. Anisotropic Delaunay meshes of surfaces. *ACM Transactions on Graphics* (2014).

[2] Chernikov, A. N., and Chrisochoides, N. P. Generalized insertion region guides for Delaunay mesh refinement. *SIAM Journal on Scientific Computing 34*, 3 (2012), A1333–A1350.

[3] Chew, L. P. Guaranteed-quality delaunay meshing in 3d (short version). In *Proceedings of the thirteenth annual symposium on computational geometry* (1997), ACM, pp. 391–393.

[4] Erten, H., and Üngör, A. Quality Triangulations with Locally Optimal Steiner Points. *SIAM J. Sci. Comp. 31*, 3 (2009), 2103–2130.

[5] Foteinos, P., and Chrisochoides, N. 4D Space-Time Delaunay Meshing for Medical Images. In *Proceedings of the 22nd International Meshing Roundtable*. Springer, 2014, pp. 223–240.

[6] Freitag, L. A., and Ollivier-Gooch, C. Tetrahedral Mesh Improvement using Swapping and Smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[7] Frey, P. J., and Alauzet, F. Anisotropic Mesh Adaptation for CFD Computations. *Computer methods in applied mechanics and engineering 194*, 48 (2005), 5068–5082.

[8] Klingner, B. M. *Tetrahedral Mesh Improvement*. PhD thesis, Berkeley, California, 2008.

[9] Klingner, B. M., and Shewchuk, J. R. Aggressive Tetrahedral Mesh Improvement. In *Proceedings of the 16th International Meshing Roundtable*, M. L. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 3–23.

[10] Labelle, F., and Shewchuk, J. R. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the nineteenth annual symposium on Computational geometry* (2003), ACM, pp. 191–200.

[11] Lévy, B., and Bonneel, N. Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In *Proceedings of the 21st International Meshing Roundtable*. Springer, 2013, pp. 349–366.

[12] Linardakis, L., and Chrisochoides, N. Delaunay decoupling method for parallel guaranteed quality planar mesh refinement. *SIAM Journal on Scientific Computing 27*, 4 (2006), 1394–1423.

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

[14] Lu, Q., Shephard, M. S., Tendulkar, S., and Beall, M. W. Parallel curved mesh adaptation for large scale high-order finite element simulations. In *Proceedings of the 21st International Meshing Roundtable*. Springer, 2013, pp. 419–436.

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

[16] Persson, P. O., and Peraire, J. Curved mesh generation and mesh refinement using lagrangian solid mechanics. *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit* (2009).

[17] Pirzadeh, S. Z., and Zagaris, G. Domain decomposition by the advancing-partition method for parallel unstructured grid generation. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting. AIAA-American Institute of Aeronautics and Astronautics* (2009).

[18] Rebay, S. Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal of Computational Physics 106*, 1 (1993), 125 – 138.

[19] Roca, X., Gargallo-Peiró, A., and Sarrate, J. Defining quality measures for high-order planar triangles and curved mesh generation. In *Proceedings of the 20th International Meshing Roundtable*. Springer, 2012, pp. 365–383.

[20] Üngör, A. Off-centers: A New Type of Steiner Points for Computing Size-optimal Guaranteed-quality Delaunay Triangulations. *Computational Geometry: Theory and Applications 42*, 2 (2009), 109–118.

[21] Üngör, A., and Sheffer, A. Pitching tents in space-time: Mesh generation for discontinuous Galerkin method. *International Journal of Foundations of Computer Science 13*, 02 (2002), 201–221.