Polygonal surface mesh simplification algorithm

1 Abstract

t This document proposes to develop a novel approach for generating a polygonal mesh representation of the surface shell of a 3D object or structure. Building upon the principles established in the 2D algorithm known as Polylla, our method adapts and extends this algorithm to operate in three dimensions. Initially, we select a suitable software tool capable of generating a 3D surface Delaunay triangulation from a given set of points. Following this, we undertake the task of designing and implementing a new algorithm to effectively operate on 3D surface meshes. Our algorithm will be designed to address the unique challenges posed by 3D geometry, where triangular faces may not necessarily lie on the same plane as in the 2D scenario. Consequently, our refined approach incorporates new criteria for selecting a set of triangles and eliminating redundant vertices to simplify the original triangular mesh. By leveraging advancements in computational geometry and mesh processing techniques, we aim to produce a robust and efficient solution for creating simplified polygonal mesh representations of complex 3D objects.

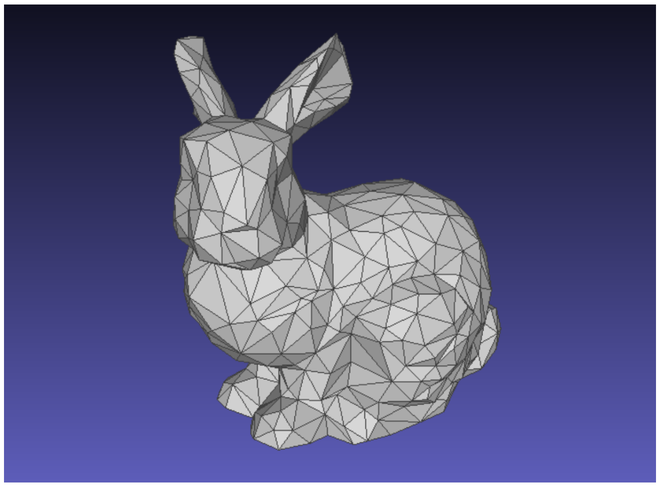

Figure 1: Example of a 3D surface triangulation mesh of a bunny model featuring 1,000 faces and 1,623 vertices. This mesh was generated by the Visual Computing Lab and visualized using MeshLab [8].

2 Introduction

Surface polygonal meshes describe, through a physical representation, the exterior shell of a 3D structure defined by a set of points. This structure could represent any 3D object and thus surface polygonal meshes have multiple applications. Among these are medical imaging, topological representation of a terrain, the representation of objects in video games, aerodynamic simulations, etc. Currently, various publicly

available codes can generate 3D surface representations through the use of triangular (see Figure 1) and quad meshes (see Figure 2). However, its difficult to find publicly available implementations for more general polygonal mashes. We are especially interested in the case where we want a mesh formed by polygons of an arbitrary number of sides because these meshes are more flexible for simplification.[1]. This simplification will allow us both to describe key features of the 3D topology of an object and to save data (comparing with a triangular or quad mesh). As a result, the efficiency of any algorithm that processes a mesh will improve when the objects mesh is simplified (see Figure 2).
For 2D structures, a new simplification polygonal algorithm called Polylla has been introduced [25]. This algorithm starts with an initial triangulation derived from a set of 2D sampling points and segments and processes it to produce a simplified polygonal mesh, typically averaging around 8 vertices per polygon. Our objective is to write a new algorithm that extend and expand Polylla algorithm to 3D surfaces. This new algorithm will begin with a predefined 3D surface (shell) triangulation and generate a polygonal mesh that simplifies the original triangulation. Unlike standard polygonal meshes like Voronoi, which produce convex polygons, the Polylla algorithm might generate non-convex polygons from an initial triangulation. This key difference entails a comparative analysis of both approaches. Additionally, while Polylla can be applied to surface meshes, it requires all faces to remain coplanar. To address this constraint, we will develop a new joining criterion that ensures coplanarity, enabling effective use of our implementation for surface meshes. As an alternative, we will explore using non-coplanar terminal edge regions as final polygons. This approach allows us to maintain a low polygon count while avoiding the oversimplification of certain areas of the object. For non-coplanar polygons, we need to develop a method of storage that enables easy recovery of the original triangulation, ensuring the preservation of the object's geometry.

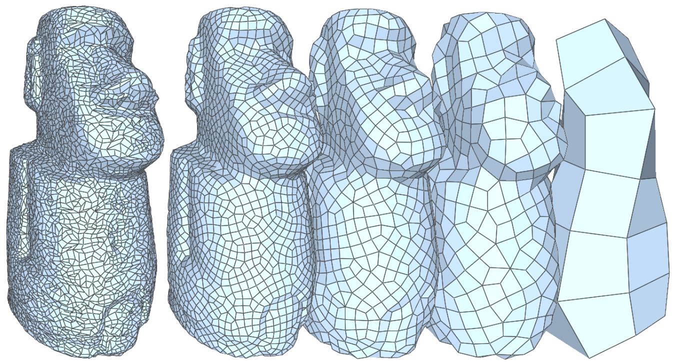

Figure 2: Simplification algorithm from [27] applied for coarsening a "Moai mesh".

3 Background

Given that as in the case of [25] our intention to employ a precomputed Delaunay Triangulation (e.g., utilizing CGAL [28]), as a starting point of our algorithm, it is pertinent to provide some contextual groundwork. The Delaunay triangulation, serving as the dual graph of the Voronoi diagram, maximizes the minimum angle of the triangles within the triangulation, thereby mitigating the formation of sliver

[1]Simplification is the process of transforming a given polygonal mesh into another mesh with fewer faces, edges, and vertices [17].

triangles[2]. The construction of Delaunay triangulation for 3D surfaces is more complex compared to its 2D counterpart. One approach involves constructing the triangulation in the 2D parametric space and then projecting it onto the surface using a mapping function [32]. A more comprehensive method for constructing a 3D surface Delaunay triangulation (or restricted Delaunay triangulation) is proposed by [2, 6][3]. This algorithm is designed to ensure sufficient sampling to yield a reliable mesh approximation of the original 3D surface while mirroring the triangulation properties observed in the 2D scenario. Another approach, presented by [14, 22], allows for obtaining a 3D surface Delaunay triangulation from any arbitrary triangulation, provided that certain topological conditions are maintained. For 3D surface meshes, the Voronoi diagram can be derived directly from the Delaunay triangulation or from an arbitrary triangulation with a set of seed sites on the surface [30, 31]. These seed sites create a partition map, with each partition enabling the construction of a local Voronoi diagram that approximates the Voronoi diagram around each site.

Since our proposal is based on the Polylla algorithm, we provide in this paragraph a brief revision of this tool. Polylla is an algorithm that starts with a 2D triangulation to derive a polygonal mesh. Before delving into its description, it's imperative to elucidate the concepts of longest-propagation path and terminal edge regions. A longest-propagation path, denoted as Leep(t_0) [24], constitutes a contiguous sequence $t_0, \ldots, t_l$ of triangles. Along this sequence, for any two consecutive triangles t_{i-1} and t_i, the longest edge of t_i surpasses that of t_{i-1}, with t_{i-1} and t_i being neighbours by sharing t_{i-1}'s longest edge. Terminal edge regions represent the union of two or more longest-edge propagation paths, sharing a triangle with a local-maximum longest edge within the region (see Figure 3). Employing these definitions, Polylla proceeds by initially identifying the terminal edge regions of the initial triangulation. It subsequently eliminates the edges of triangles within these regions, resulting in a mesh composed of polygons bounded by the borders of the identified terminal edge regions (see Figure 3). Given that Polylla operates based on a Delaunay triangulation, mesh comparisons were conducted with Voronoi meshes in [25], where the Voronoi mesh denotes the dual polygonal mesh derived from a Delaunay triangulation. The meshes produced by Polylla exhibit greater simplification compared to Voronoi meshes, containing approximately half the number of polygons and points [25].

Our implementation centres on mesh simplification, a well-studied area for both triangular and quad meshes (e.g., [5, 27]). Triangle mesh simplification, a mature technology since the 1990s, is integrated into tools like Maya, Blender, and MeshLab [9]. These methods use adaptive meshing to reduce triangles while maintaining shape fidelity [7, 23], employing iterative local modifications for desired Levels of Detail (LOD) and Continuous LOD (CLOD) models. Hoppe et al. [18] introduced local operators such as edge flips to enhance tessellation quality, guided by an objective function. An algorithm for simplifying 3D surface triangulations is the one proposed by [15, 16] where the simplification is performed through iterative contractions of vertex pairs. A pair contraction described by $(v_1, v_2) \to \bar{v}$ is obtained by selecting $\bar{v}$ that minimizes the quadratic error metrics and in this way the results obtained characterize the overall shape of the surface.

Quad mesh simplification, a more recent development, presents unique challenges. Local operations like quad-diagonal collapse [10, 11, 20] maintain the quad structure but need larger area operations for overall quality. Techniques like poly-chord collapse [4] remove entire lines of quads to preserve regularity. Global operations are impractical due to issues with discontinuous resolution changes and selective decimation, making local operations preferable for out-of-core adaptations. Other methods [10, 26] advocate for steering collapses within subregions and breaking down poly-chord collapses into

[2]A triangle characterized by at least one extremely acute angle, resulting in a long/thin shape

[3]`https://doc.cgal.org/latest/Surface_mesher/index.html`

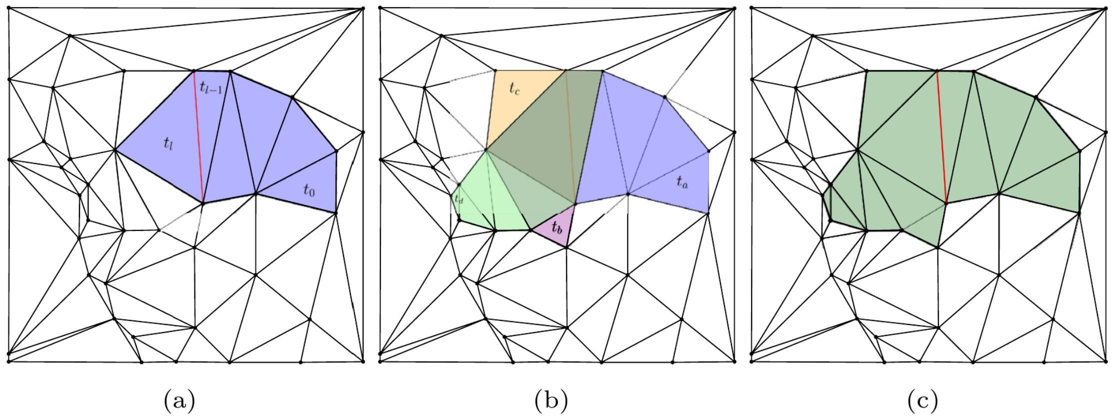

Figure 3: Description of how to get a polygonal mesh from a triangular mesh through Polylla (from [2]). The polygon colored in green in the right panel (c) has been obtained by the union of 4 Leeps. The first one is in the left panel (a) in blue, the other 3 are shown in the middle panel (b) in orange, light green and pink, respectively.

smaller steps, creating a fine-grained, local-only framework. Interleaving tangent space smoothing with each iteration improves quality and helps select subsequent operations.

Quad-remeshing focuses on creating a new, high-quality mesh from the input shape, not necessarily reducing complexity. Techniques [1, 13, 12, 29, 19, 21] use principal curvatures and mesh parametrization. This global operation faces challenges with noise and complex issues like parametrization and curvature direction but produces superior quality meshes ideal for CAD models by enforcing vertex valency correspondence with Gaussian curvature [3, 21]. Overall, local, greedy simplification performs comparably to global methods like [3], offering advantages in adaptiveness and robustness. In [27], a comprehensive algorithm for simplifying quad meshes is presented. The implementation described transforms a high-complexity mesh into a low-complexity one through iterative local operations, preserving mesh structure until a termination criterion is met. Primary operations include edge and vertex rotations, diagonal collapse (see Figure 4), and smoothing (both local and global), aimed at reducing the number of squares in the mesh.

4 Problem Statement

To apply Polylla to a 3D surface triangular mesh, several new constraints must be met. The foremost requirement is to ensure that the transformation from a triangular to a polygonal mesh preserves flat surfaces. This necessitates selecting terminal edge regions based on a new criterion. For meshes representing objects with multiple flat faces (e.g., polyhedra), this criterion must prevent the merging of terminal edge regions that span distinct faces. Additionally, the new join criterion typically results in polygons with fewer triangles compared to the 2D case. Therefore, to achieve effective mesh simplification, we need to establish further criteria, such as eliminating vertices in sets of neighbouring, nearly coplanar vertices. Alternatively, if we relax the requirement for coplanar polygonal faces, we should explore the possibility of a new data structure where each polygon consists of triangles that are not necessarily coplanar.

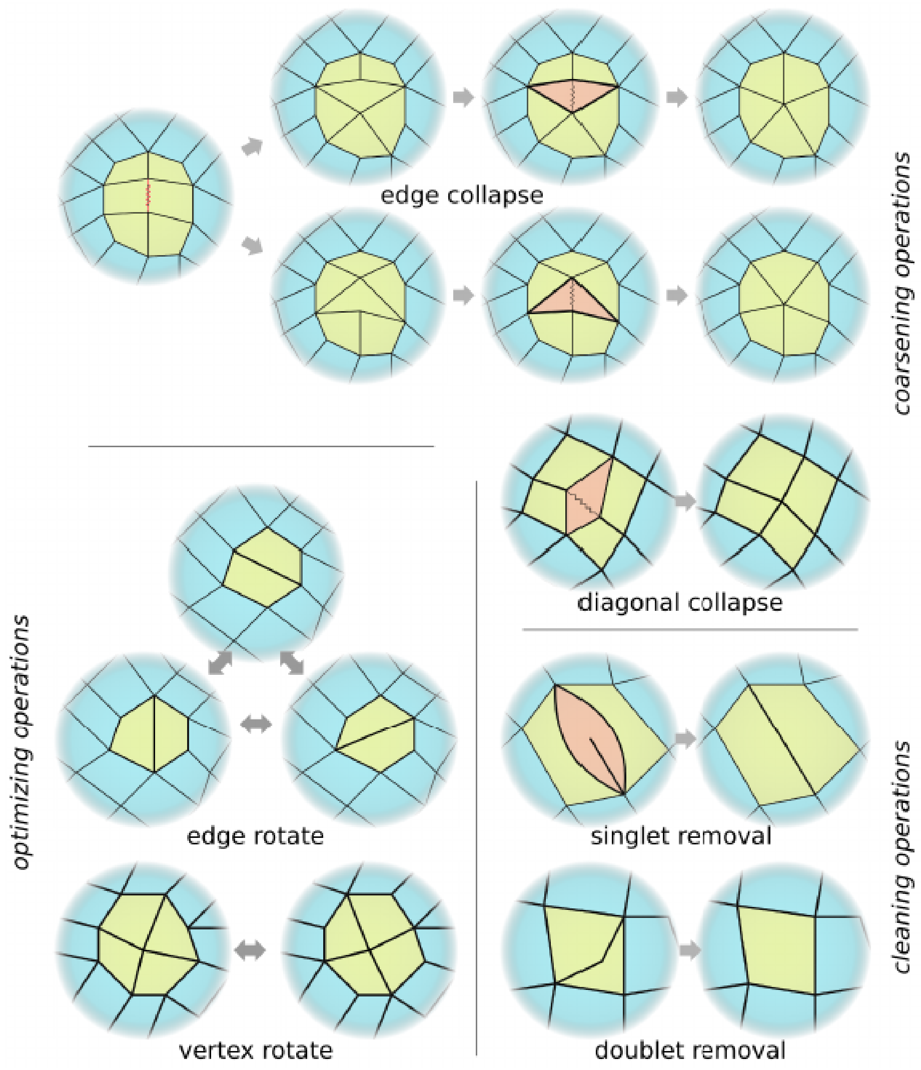

Figure 4: Local operations used to obtain quad mesh simplifications (from [27]).

5 Research questions

Is it possible to construct a simplification algorithm that represents surfaces using general polygons with fewer vertices, arcs, and polygons than triangulations, while allowing for a better representation of their shape? Which criteria can be used to eliminate points from the initial triangulation?

6 Hypothesis

To address our main research question, we hypothesize that meshes composed of general polygons allow for the representation of surfaces with fewer polygons, edges, and vertices compared to triangulation or quadrilateral meshes. This approach aims to improve precision and reduce the utilized space.

7 Objectives

The main objective of this project is to obtain an algorithm that simplify a 3D surface triangular mesh by selecting terminal edge regions and eliminating redundant vertices. In order to accomplish this we have the following secondary objectives:

1. Establish a criterion for joining triangles (to form polygons) from a triangular mesh that maintains or does not maintain coplanarity, considering the floating-point problem.
2. Establish criteria for eliminating surface vertices.
3. Develop an algorithm that, using the criteria established in the previous two points, simplifies an initial triangulated 3D surface mesh.
4. Compare the new polygonal algorithm with the CGAL and Meshlab simplification algorithm in terms of the number of polygons, edges, points, and the accuracy of the surface representation.
5. Extend the visualization tool Camaron-web[4] to display non-convex polygons.

8 Methodology

In the first stage, we will select an appropriate tool to generate the initial triangulation. The primary candidate for this purpose is CGAL [28], which offers a Delaunay 3D surface mesh that is based on the algorithm of [2, 6][3]. The second stage involves evaluating Camaron-web's ability to display non-convex polygons. This will include creating a basic scenario where a surface mesh is modified to incorporate non-convex polygons. We will then use this modified mesh to assess the performance of the Camaron-web visualization tool. Given that visualization is crucial for debugging our algorithms, the aim of these tests is to determine how effectively this tool can visualize the polygonal meshes developed throughout this thesis.
After completing the assessment of our visualization tool, our next step will be to establish criteria for simplifying the original triangular mesh. This includes defining rules for merging triangles during the construction of terminal edge regions and for eliminating vertices. Once these criteria are set, the core phase of the project will involve implementing an algorithm to construct a polygonal mesh from an initial

[4]`https://github.com/DiegoZQ/camaron-web`

3D surface triangulation. We will begin by exploring the methodology for acquiring terminal edge regions, with a particular focus on scenarios involving non-convex polygons. Furthermore, we will tackle floating-point precision issues, particularly in scenarios involving coplanar faces. Following this, we will conduct a comprehensive analysis of the existing Polylla codebase[5] to restructure point representations from 2D to 3D as the default. Simultaneously, efforts will focus on adapting 2D terminal edge regions within the codebase to support 3D surface meshes. Additionally, we will analyze and implement modifications to ensure the existing 2D half-edge data structure functions correctly with surface meshes. After completing these structural modifications, the triangulation constructor code in Triangulation.cpp will be updated to handle 3D surface mesh inputs. A key challenge is that the 2D constructor initially builds interior half-edges before constructing boundary edges. Since surface meshes lack boundaries, it is essential to determine whether this step is necessary or if the entire constructor requires revision. Additionally, the criteria for selecting terminal edge regions are anticipated to be more limited than in 2D cases, as the triangles in each terminal edge region are not necessarily coplanar. Given this limitation, we also expect to use other criteria to simplify the mesh, such as vertex elimination in cases where there is a set of neighbouring vertices that are nearly coplanar, vertex decimation via pair contraction (e.g.,[15, 16]) or local operations (as in [27]). We will also evaluate the use (or construction) of data structures where the faces that form a polygon are defined by a set of triangles that are not necessarily coplanar.

Since our algorithm involves 3D adaptations of Polylla and CGAL codes, we plan to implement it in C++. However, we will also explore the possibility of adopting an alternative approach, potentially leveraging the capabilities of the OpenMesh Python library.[6] Upon completion of our algorithm, we will proceed to conduct rigorous testing of our implementation against established standards such as the Voronoi diagram. Of particular interest will be the comparison between our novel implementation and a Voronoi mesh, utilizing a renowned 3D mesh example, notably the ”Stanford rabbit” dataset.[7] We anticipate that, akin to observations with Polylla [25], under equivalent initial conditions, our new implementation should yield a similar structural representation employing fewer polygons compared to a Voronoi mesh. Additionally, we will assess the execution time of ouralgorithm relative to that of the Voronoi algorithm. For the second set of tests, we will utilize the default CGAL simplification tools[8] to simplify the original triangular mesh used by our algorithm. Our aim is to compare our algorithm with the CGAL simplification algorithm in terms of the number of polygons, edges, points, and the accuracy of the surface representation.

9 Expected Results

We expect to design a new algorithm that from an initial 3d surface mesh can obtain a simplify a mesh of polygons with an arbitrary number of edges. The resulting implementation will reduce the size a mesh by keeping the essential geometric characteristics of the represented object. The algorithms and the results obtained in this thesis will be presented in the Siam International Meshing Rountable Workshop[9].

[5] `https://github.com/ssalinasfe/Polylla-Mesh-DCEL`

[6] `https://www.graphics.rwth-aachen.de/media/openmesh_static/Documentations/OpenMesh-6.2-Documentation/a00016.html`

[7] `https://graphics.stanford.edu/data/3Dscanrep/`

[8] `https://doc.cgal.org/latest/Surface_mesh_simplification/index.html`

[9] `https://internationalmeshinground table.com/`

References

[1] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy & Mathieu Desbrun (2003): *Anisotropic polygonal remeshing*. In: *ACM SIGGRAPH 2003 Papers*, pp. 485–493.

[2] Jean-Daniel Boissonnat & Steve Oudot (2005): *Provably good sampling and meshing of surfaces*. *Graphical Models* 67(5), pp. 405–451, doi:https://doi.org/10.1016/j.gmod.2005.01.004. Available at `https://www.sciencedirect.com/science/article/pii/S1524070305000056`. Solid Modeling and Applications.

[3] David Bommes, Henrik Zimmer & Leif Kobbelt (2009): *Mixed-integer quadrangulation*. *ACM transactions on graphics (TOG)* 28(3), pp. 1–10.

[4] Michael J Borden, Steven E Benzley & Jason F Shepherd (2002): *Hexahedral Sheet Extraction.* In: *IMR*, pp. 147–152.

[5] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez & B. Levy (2010): *Polygon Mesh Processing*. CRC Press. Available at `https://books.google.cl/books?id=AuXqBgAAQBAJ`.

[6] L. Paul Chew (1993): *Guaranteed-quality mesh generation for curved surfaces*. In: *Proceedings of the Ninth Annual Symposium on Computational Geometry*, SCG '93, Association for Computing Machinery, New York, NY, USA, p. 274–280, doi:10.1145/160985.161150. Available at `https://doi.org/10.1145/160985.161150`.

[7] P. Cignoni, C. Montani & R. Scopigno (1998): *A comparison of mesh simplification algorithms*. *Computers & Graphics* 22(1), pp. 37–54, doi:https://doi.org/10.1016/S0097-8493(97)00082-4. Available at `https://www.sciencedirect.com/science/article/pii/S0097849397000824`.

[8] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli & Guido Ranzuglia (2008): *MeshLab: an Open-Source Mesh Processing Tool.* In Vittorio Scarano, Rosario De Chiara & Ugo Erra, editors: *Eurographics Italian Chapter Conference*, The Eurographics Association, doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[9] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli & Guido Ranzuglia (2008): *MeshLab: an Open-Source Mesh Processing Tool.* In Vittorio Scarano, Rosario De Chiara & Ugo Erra, editors: *Eurographics Italian Chapter Conference*, The Eurographics Association, doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[10] Joel Daniels, Claudio T Silva & Elaine Cohen (2009): *Localized quadrilateral coarsening*. In: *Computer Graphics Forum*, 28, Wiley Online Library, pp. 1437–1444.

[11] Joel Daniels, Cláudio T Silva, Jason Shepherd & Elaine Cohen (2008): *Quadrilateral mesh simplification*. *ACM transactions on graphics (TOG)* 27(5), pp. 1–9.

[12] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci & John C Hart (2006): *Spectral surface quadrangulation*. In: *Acm siggraph 2006 papers*, pp. 1057–1066.

[13] Shen Dong, Scott Kircher & Michael Garland (2005): *Harmonic functions for quadrilateral remeshing of arbitrary manifolds*. *Computer aided geometric design* 22(5), pp. 392–423.

[14] Ramsay Dyer, Hao Zhang & Torsten Möller (2007): *Delaunay mesh construction*. In: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, Eurographics Association, Goslar, DEU, p. 273–282.

[15] Michael Garland & Paul S. Heckbert (1997): *Surface simplification using quadric error metrics*. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., USA, p. 209–216, doi:10.1145/258734.258849. Available at `https://doi.org/10.1145/258734.258849`.

[16] Michael Garland & Paul S. Heckbert (2023): *Surface Simplification Using Quadric Error Metrics*, 1 edition. Association for Computing Machinery, New York, NY, USA. Available at `https://doi.org/10.1145/3596711.3596727`.

[17] Craig Gotsman, Stefan Gumhold & Leif Kobbelt (2002): *Simplification and Compression of 3D Meshes*, pp. 319–361. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-662-04388-2_12. Available at `https://doi.org/10.1007/978-3-662-04388-2_12`.

[18] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald & Werner Stuetzle (1993): *Mesh optimization*. In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, Association for Computing Machinery, New York, NY, USA, p. 19–26, doi:10.1145/166117.166119. Available at `https://doi.org/10.1145/166117.166119`.

[19] Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt & Hujun Bao (2008): *Spectral quadrangulation with orientation and alignment control*. In: *ACM SIGGRAPH Asia 2008 papers*, pp. 1–9.

[20] Paul Kinney (1997): *Cleanup: Improving quadrilateral finite element meshes*. In: *6th International Meshing Roundtable*, pp. 437–447.

[21] Yu-Kun Lai, Leif Kobbelt & Shi-Min Hu (2008): *An incremental approach to feature aligned quad dominant remeshing*. In: *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pp. 137–145.

[22] Yong-Jin Liu, Chun-Xu Xu, Dian Fan & Ying He (2015): *Efficient construction and simplification of Delaunay meshes*. *ACM Trans. Graph.* 34(6), doi:10.1145/2816795.2818076. Available at `https://doi.org/10.1145/2816795.2818076`.

[23] David Luebke (2003): *Level of detail for 3D graphics*. Morgan Kaufmann.

[24] María-Cecilia Rivara (1997): *New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations*. *International journal for numerical methods in Engineering* 40(18), pp. 3313–3324.

[25] Sergio Salinas-Fernández, Nancy Hitschfeld-Kahler, Alejandro Ortiz-Bernardin & Hang Si (2022): *POLYLLA: polygonal meshing algorithm based on terminal-edge regions*. *Engineering with Computers* 38(5), pp. 4545–4567.

[26] Jason F Shepherd, Mark W Dewey, Adam C Woodbury, Steven E Benzley, Matthew L Staten & Steven J Owen (2010): *Adaptive mesh coarsening for quadrilateral and hexahedral meshes*. *Finite Elements in Analysis and Design* 46(1-2), pp. 17–32.

[27] Marco Tarini, Nico Pietroni, Paolo Cignoni, Daniele Panozzo & Enrico Puppo (2010): *Practical quad mesh simplification*. In: *Computer Graphics Forum*, 29, Wiley Online Library, pp. 407–418.

[28] The CGAL Project (2023): *CGAL User and Reference Manual*, 5.6 edition. CGAL Editorial Board. Available at `https://doc.cgal.org/5.6/Manual/packages.html`.

[29] Yiying Tong, Pierre Alliez, David Cohen-Steiner & Mathieu Desbrun (2006): *Designing quadrangulations with discrete harmonic forms*. In: *Eurographics symposium on geometry processing*.

[30] Shiqing Xin, Pengfei Wang, Rui Xu, Dongming Yan, Shuangmin Chen, Wenping Wang, Caiming Zhang & Changhe Tu (2022): *SurfaceVoronoi: Efficiently Computing Voronoi Diagrams Over Mesh Surfaces with Arbitrary Distance Solvers*. *ACM Trans. Graph.* 41(6), doi:10.1145/3550454.3555453. Available at `https://doi.org/10.1145/3550454.3555453`.

[31] Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun & Wenping Wang (2009): *Isotropic remeshing with fast and exact computation of Restricted Voronoi Diagram*. In: *Proceedings of the Symposium on Geometry Processing*, SGP '09, Eurographics Association, Goslar, DEU, p. 1445–1454.

[32] Yao Zheng, Roland W. Lewis & David T. Gethin (1996): *Three-dimensional unstructured mesh generation: Part 2. Surface meshes*. *Computer Methods in Applied Mechanics and Engineering* 134(3), pp. 269–284, doi:https://doi.org/10.1016/0045-7825(95)00917-5. Available at `https://www.sciencedirect.com/science/article/pii/0045782595009175`.