Lecture Notes in Computational Science and Engineering

Volume 143

Vladimir A. Garanzha • Lennard Kamenski •
Hang Si
Editors

Numerical Geometry, Grid Generation and Scientific Computing

Proceedings of the 10th International Conference, NUMGRID 2020 / Delaunay 130, Celebrating the 130th Anniversary of Boris Delaunay, Moscow, Russia, November 2020

Springer

Editors
Vladimir A. Garanzha
Dorodnicyn Computing Centre, Federal
Research Center of Informatics and Control
Russian Academy of Sciences Moscow
Moscow, Russia

Lennard Kamenski
Berlin, Germany

Hang Si
Weierstrass Institute for Applied Analysis
and Stochastics (WIAS)
Berlin, Germany

ISSN 1439-7358 ISSN 2197-7100 (electronic)
Lecture Notes in Computational Science and Engineering
ISBN 978-3-030-76797-6 ISBN 978-3-030-76798-3 (eBook)
https://doi.org/10.1007/978-3-030-76798-3

Mathematics Subject Classification (2010): 65-xx (65Dxx, 65Fxx, 65Kxx, 65Lxx, 65Mxx, 65Nxx, 65Yxx, 65Zxx), 30Cxx, 30Fxx, 49Mxx, 52Cxx, 53xx

Dedicated to
Boris Nikolayevich Delaunay *(1890–1980)*
on the occasion
of his 130th birthday

Foreword

This volume presents the proceedings of the NUMGRID 2020/Delaunay 130 International Conference dedicated to the 130th birthday of B. N. Delaunay (1890–1980). Since Boris Nikolayevich was my teacher, I will take this nice opportunity to have another look at certain moments in the life and work of this unique personality.

Delaunay triangulation, Delaunay partition, and their theory is the most important part of his research from an application point of view. This is, however, merely one facet in his multifaceted work, a facet to which B. N. Delaunay came not by chance. There is a deeper meaning in the fact that the name of Delaunay is forever inscribed in science next to that of G. F. Voronoi, a prominent representative of the St. Petersburg school of number theory. Even more so, because Georgy Feodosievich was a close friend of the Delaunay family. As a teenager, Boris often witnessed his father's late evening conversations with Voronoi.

Voronoi was not and could not be Delaunay's scientific supervisor: he died unexpectedly in 1908, the year Boris entered university. Nevertheless, his influence on Delaunay was considerable. Delaunay's breakthrough results on the cubic Diophantine equations, which are, by his own admission, his most outstanding work, used the famous Voronoi algorithm for finding fundamental units in cubic fields. This celebrated work of Voronoi once made a stunning impression on A. A. Markov (Voronoi's teacher) himself.

In the early 1920s, Delaunay was invited by A. A. Markov to the Petrograd[1] University as a professor and wrote a remarkable paper revealing the geometrical essence of the Voronoi algorithm. In the late 1920s, Delaunay published a major work on 4-dimensional parallelohedra, where he continued the research of H. Minkowski and G. Voronoi on the theory of parallelohedra. At the same time, Delaunay elegantly introduced the important concepts of the (r, R)-system and the L-partition corresponding to this system and published an extensive paper *"Geometry of positive quadratic forms"*. The concepts laid down in Delaunay's works in the 1920s and 1930s, influenced by Voronoi's work, proved to be useful

[1]The name of St. Petersburg in 1914–1924.

in computational geometry, crystallography, structural chemistry, biology, and other fields. As for the terms (r, R)-system and L-partition, much later, in the second half of the twentieth century, the professional community abandoned them in favor of the terms *Delaunay set* and *Delaunay partition*, largely thanks to H. S. M. Coxeter and C. A. Rogers.

As for the NUMGRID 2020 conference, I would like to acknowledge the high professional level of the participants and thank its organizers Vladimir Garanzha, Hang Si, and Lennard Kamenski.

Steklov Mathematical Institute RAS, Moscow, Russia Nikolay Dolbilin
February 2021

Preface

This volume presents a selection of papers presented at the 10th International Conference on Numerical Geometry, Grid Generation, and Scientific Computing celebrating the 130th anniversary of B. N. Delaunay (NUMGRID 2020/Delaunay 130), held November 25–27, 2020. The conference is bi-annual (since 2002) and it is one of the well-known international conferences in the area of mesh generation. The main topic of this conference, grid (mesh) generation, is about how to create a geometric discretization of a given domain. It is an indispensable tool for solving field problems in nearly all areas of applied mathematics.

The book includes an overview of the current progress in numerical geometry, grid generation, and adaptation in terms of mathematical foundations, algorithm and software development, and applications. In focus are the Voronoi-Delaunay theory and algorithms for tilings and partitions, mesh deformation and optimization, equidistribution principle, error analysis, discrete differential geometry, duality in mathematical programming and numerical geometry, mesh-based optimization and optimal control methods, iterative solvers for variational problems, as well as algorithm and software development. The applications of the discussed methods are multidisciplinary and include problems from mathematics, physics, biology, chemistry, material science, and engineering.

The presented 25 papers were selected from 31 submissions. The main selection criteria are based on the recommendations of anonymous peer reviews from experts of the corresponding fields as well as the presentation of the paper at the conference. All accepted papers are revised according to the comments of reviewers and the program committee.

The organizers would like to thank all who submitted papers and all who helped to evaluate the contributions by providing reviews for the submissions. The reviewers' names are acknowledged in the following pages. The organizers would like to thank all participants of NUMGRID for making it a successful and interesting experience.

Moscow, Russia	Vladimir A. Garanzha
Berlin, Germany	Lennard Kamenski
Berlin, Germany	Hang Si
February 2021	

Contents

Conference Organization

Organizers

Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia
http://www.ccas.ru

Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany
https://www.wias-berlin.de

Lennard Kamenski, Berlin, Germany
https://gitlab.com/lkamenski

Organizing Committee

V. A. Garanzha (Chair)
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

L. Kamenski (Vice chair)
Berlin, Germany

A. I. Belokrys-Fedotov
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

I. E. Kaporin
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

L. N. Kudryavtseva
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Yu. O. Trusova
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

I. A. Zonn
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Program Committee

Yu. G. Evtushenko (Chair)
Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

H. Si (Vice Chair)
Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany

A. Belyaev
Heriot-Watt University, Edinburgh, UK

H. Borouchaki
Inria Paris-Rocquencourt, France

N. P. Dolbilin
Steklov Mathematical Institute RAS, Moscow, Russia

V. P. Dymnikov
Institute of Numerical Mathematics RAS, Moscow, Russia

H. Edelsbrunner
Institute of Science and Technology, Klosterneuburg, Austria

S. K. Godunov
Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia

R. Jain
Argonne National Laboratory, Lemont, IL, USA

V. F. Kuropatenko
All-Russian Scientific Research Institute of Technical Physics, Snezhinsk, Russia

P. Laug
Inria Paris-Rocquencourt, France

N. Lei
Dalian University of Technology, Dalian, China

V. D. Liseikin
Institute of Computing Technologies RAS, Novosibirsk, Russia

Yu. V. Nesterenko
Moscow State University, Moscow, Russia

R. V. Polozov
Institute of Cell Biophysics, Puschino, Russia

X. Roca
Barcelona Supercomputing Center, Barcelona, Spain

D. V. Sokolov
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

S. K. Vodopyanov
Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia

Web Site

http://www.ccas.ru/gridgen/numgrid2020

Reviewers

Igor Baburin Technische Universität Dresden, Dresden, Germany

Pavel Bakhvalov Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

Alexander Belyaev Heriot-Watt University, Edinburgh, UK

Matthew Bright University of Liverpool, Liverpool, UK

Michail Botchev Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

Andrey Chernikov Old Dominion University, Norfolk, VA, USA

Kristian Debrabant University of Southern Denmark, Odense, Denmark

Vladimir A. Garanzha Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Alexander Golikov Dorodnicyn Computing Center of FRC CSC RAS, Moscow, Russia

Roberto Grosso Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Xianfeng Gu Stony Brook University, Stony Brook, NY, USA

Zhenqun Guan Dalian University of Technology, Dalian, China

Ronald D. Haynes Memorial University of Newfoundland, St. John's, NL, Canada

Nancy Hitschfeld University of Chile, Santiago, Chile

Rajeev Jain Argonne National Laboratory, Lemont, IL, USA

George Kamenev (The author and reviewer of the NUMGRID conferences, passed away on November 3rd 2020.) Federal Research Center of Informatics and Management RAS, Moscow, Russia

Lennard Kamenski Berlin, Germany

Jiří Kosinka Rijksuniversiteit Groningen, Groningen, The Netherlands

Na Lei Dalian University of Technology, Dalian, China

Shu-Jie Li Beijing Computational Science Research Center, Beijing, China

Alexander Linke Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany

Vladimir Liseikin Institute of Computational Technologies, Siberian Branch RAS, Novosibirsk, Russia

Yang Liu Microsoft Research Asia, Beijing, China

Xiaoming Liu Cadence Design Systems, Inc., San Jose, CA, China

Vassily Manturov Moscow Institute of Physics and Technology, Moscow, Russia

Yulong Pan University of California, Berkeley, CA, USA

Per-Olof Persson University of California, Berkeley, CA, USA

Egon Schulte Northeastern University, Boston, MA, USA

Alexander Skovpen NUMECA International, Brussels, Belgium

Kirill Terekhov Marchuk Institute of Numerical Mathematics RAS, Moscow, Russia

Vladimir Titarev Dorodnicyn Computing Center FRC CSC RAS, Moscow, Russia

Sergey Utyuzhnikov The University of Manchester, Manchester, UK

Yuri Vassilevski Marchuk Institute of Numerical Mathematics RAS, Moscow, Russia

Krassimira Vlachkova Sofia University St. Kliment Ohridski, Sofia, Bulgaria

Rhaleb Zayer Max-Planck-Institut für Informatik, Saarbrücken, Germany

Paul Zegeling Universiteit Utrecht, Utrecht, The Netherlands

Xiaopeng Zheng Dalian University of Technology, Dalian, China

Victor Zhukov Keldysh Institute of Applied Mathematics RAS, Moscow, Russia

Daniel Zint Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Hexahedral Mesh Generation Using Voxel Field Recovery

Alexander Sergeevich Karavaev and Sergey Petrovich Kopysov

Abstract We consider a modification of the previously developed voxel-based mesh algorithm to generate models given by a triangular surface mesh format (STL). Proposed hexahedral mesh generator belongs to the family of grid methods, and its capable to use as source data both volume and surface types of model geometry representation. To define the initial position of mesh nodes, a «signed distance field» volume data file, obtained from the STL geometry, is used. A special projection technique was developed to adapt constructed orthogonal mesh on the models boundary. It provides an approximation of sharp edges and corners and performs before running any other operations with the mesh. Finally, to improve the mesh quality, additional procedures were implemented, including boundary layers insertion, bad quality cells splitting, and optimization-based smoothing technique.

1 Introduction

The first step in the process of element mesh generation belongs to the description of the initial model geometry. Surface representation is a set of faces describing a model's boundary. This description has become the main technique applied in engineering calculations and it is widely used by the existing CAD software.

However, for some applications, such as medicine and biomodeling, the only possible way is to use voxel data obtained by a CT, MRI, or MicroCT scanner [11]. Preserving geometric corners and edges is not so critical in the case of biological tissues modeling. Usually, mesh algorithms for such problems are high performance and robust. They are based on combining pairs of Cartesian voxels grid into hexahedra followed by boundary nodes position adaptation.

At the same time, generation of hexahedral meshes of a given quality with an accurate description of an arbitrary model geometry is not a completely solved

A. S. Karavaev (✉) · S. P. Kopysov
Department of Computational Mechanics, Udmurt State University, Izhevsk, Russia
e-mail: karavaev-alexander@yandex.ru

V. A. Garanzha et al. (eds.), *Numerical Geometry, Grid Generation and Scientific Computing*, Lecture Notes in Computational Science and Engineering 143,
https://doi.org/10.1007/978-3-030-76798-3_19

task [1, 8]. In comparison with tetrahedral meshes, there is no well-developed theory and robust general-purpose software to solve the problem under discussion.

Specially for this, a sufficient number of algorithms have been developed with their advantages and disadvantages.

For example, block decomposition approaches break the model into pieces that are easier to mesh. They provide good quality cells but are semi-automatic, and therefore time-consuming to construct complex geometries. It should be mentioned that a lot of strategies have been developed to automate the block decomposition process, such as medial surface transformation, midpoint subdivision, and singularity-restricted frame fields [1].

The advancing fronts methods (sweeping, paving, plastering, etc.) generate a hex-mesh from the boundary of the surface mesh inward [9]. Such techniques have issues with robustness, especially when two different fronts are merged during processing. The same problems of reliability have the methods based on spatial twist continuum concept which is a dual representation of an all hex-mesh that defines the global connectivity constraints [6]. In recent years, many investigations have been devoted to the indirect approaches. They convert a given tetrahedral mesh to the hexahedral one by using a different type of combining techniques [3, 8]. Unfortunately, in many cases, the result meshes are hex-dominant while containing a small number of irregular polyhedra.

The grid-based or octree family of methods are robust but tends to generate poor quality elements at the boundary of the volume and to create hanging nodes arising from size transitions [1, 10]. Also, these methods incline to produce a large number of hexahedra and are highly dependent upon the orientation of the interior orthogonal grid of hex elements. The recent attempts to quality hexahedral meshing are either robust but lack the ability to align to the surface curvature and sharp features [4, 7], or produce high quality meshes but are extremely fragile.

In this article, we propose an algorithm that provides good compromise between robustness and quality meshing. We consider a modification of previously developed voxel mesh generator which was originally intended to work with volume (tomograthic) data representation [5]. Figure 1 shows the result of the algorithm for construction a multicomponent head model. Here, the boundary is reconstructed with the «Dual Contouring» algorithm [5, 11]. This approach is widely used to render tomographic images with a surface quadrangular mesh. It allows to restore the sharp features of the model contour to a certain degree. Unfortunately, in the case of STL geometry, the accuracy of the «Dual Contouring» algorithm is not sufficient. Therefore, additional options have to be considered to build hexahedral meshes of models given in STL format.

Our approach belongs to the family of grid methods. It utilizes initial surface representations and «signed distance fields» voxel data format obtained from STL file.

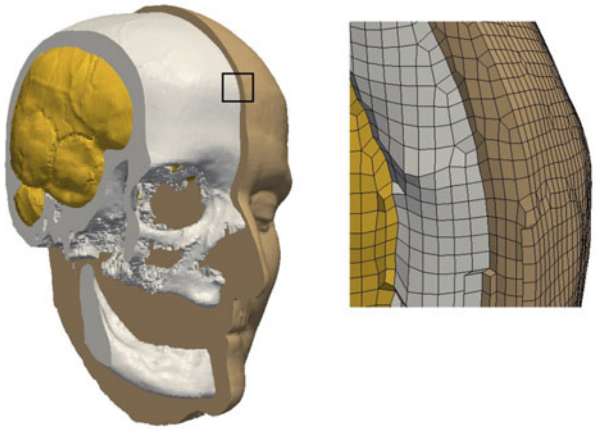
Fig. 1 Human head model with three materials. Hexahedral mesh constructed from tomograthic data by the authors' algorithm [5]

2 Initial Data Representation

As mentioned before, the proposed method can generate meshes from both surface and volumetric types of model representation.

In literature, volumetric (voxel) data $\mathcal{V}$ is defined as scalar values of an implicit function $\mathcal{F}$ on a Cartesian coordinate grid $\mathcal{V} = \mathcal{F}(i, j, k)$, where i, j, k are indexes in the form of x, y, z coordinates of Cartesian grid. The isosurface (surface of equal values) corresponding to the value α represents points of a constant value within a volume $I_{\mathcal{F}}(\alpha) = \{\, (x, y, z) \mid \mathcal{F}(x, y, z) = \alpha \,\}$.

Volumetric data may be viewed by extracting isosurfaces from the volume and rendering them as polygonal meshes or by rendering the volume directly as a block of data.

For now, the algorithm takes a standard DICOM volumetric data file obtained from tomography scans to construct meshes of biological tissues. In the case of STL-models, the «signed distance fields» (SDF) data structure is used.

Each voxel in a SDF file contains a positive or a negative shortest distance to the model surface according to its relative position. Therefore, a border of the model is set by the isosurface of value $\alpha = 0$.

Let the model Ω be bounded by a surface triangular mesh T. Then, for an arbitrary voxel $\boldsymbol{v}$, the function $\mathcal{F}$ could be written as

$$\mathcal{F}(\boldsymbol{v}) = S_{\mathcal{T}}(\boldsymbol{v}) \times \min_{t \in T} d(t, \boldsymbol{v}), \quad S_T(\boldsymbol{v}) = \begin{cases} +1, & \text{if } \boldsymbol{v} \subseteq \Omega, \\ -1, & \text{if } \boldsymbol{v} \not\subseteq \Omega, \end{cases}$$

where $d(t, \boldsymbol{v})$ denotes the minimal distance between a triangle $t \in T$ and a voxel $\boldsymbol{v} \in \mathcal{V}$.

The initial SDF voxel array is constructed from a given STL geometry by a straightforward («brute force») conversion technique. For every voxel $\boldsymbol{v} \in \mathcal{V}$ it calculates the distance to all triangles of T and takes the minimal one. Because of this, the execution time is a little bit high and it is efficient to use parallel computations. On the other hand, this method calculates the accurate Euclidean distance field without errors and has a simple implementation.

The obtained volume data file implicitly contains all information about an orthogonal hexahedral grid covering the model. Thus, a result of the mesh generation directly depends on the accuracy of voxels values in $\mathcal{V}$.

3 Hexahedral Mesh Generation

According to Schneider [10], the mesh of the model should fulfill the following requirements: each point of the model is covered by the mesh point; each edge of the model is covered by a sequence of mesh edges.

To overcome this problem, we construct the set Λ of «characteristic» edges and nodes. The set Λ is considered to be a «frame» of the model which preserves all its sharp features. The resulting hexahedral mesh $\mathcal{T}$ has to cover the elements of Λ with a continuous chain of its edges and nodes.

To construct Λ we follow Schneider's idea and don't take into account edges of triangular mesh T whose adjacent faces enclose a dihedral angle of about 180°.

In Fig. 2a and b we can see the STL geometry of a half part of the metal sleeve and its set Λ with 10 «characteristic» nodes and 15 «characteristic» edges.

The second step of mesh generation is the construction of the orthogonal structured grid with size h (Fig. 2c). The grid size h is chosen in advance, and it defines an interval (span) between voxels that will be taken to form mesh nodes. This size determines the number of cells in the resulting mesh and thus a final accuracy of the model description. The minimal value of h is 1, and it means that all voxels of $\mathcal{V}$ taking into account during mesh generation.

For every 8 neighboring voxels with $\gamma = 4$ or more positive values a new cell is constructed.

Let $\dim_z \times \dim_y \times \dim_x$ be the dimension of SDF voxel array $\mathcal{V}$ and $sgn_{\mathcal{F}}^{(x,y,z)}$ is a signum function of voxel value with indexes (x, y, z), $sgn_{\mathcal{F}}^{(x,y,z)} = sgn\big(\mathcal{F}\big(\boldsymbol{v}^{(x,y,z)}\big)\big)$, then the above process can be written as following: "Algorithm 1".

In addition to number $\gamma = 4$ of positive voxels, we also require that they have to belong to one quadrangle of a hexahedron. Such choice provides a situation when every face of a cell has at least 2 points inside the model. From tests, we established that such value $\gamma = 4$ is optimal because of a less number of additional hexahedra (in comparison with the case $\gamma < 4$, see Fig. 3a and b) and uniform size of boundary and inner cells (that is not provided in the case of $\gamma > 4$, see Fig. 3c and d).

It should be mentioned that the number of positive voxels is not strict and can be lower than 4. Also, there is an option to add any other conditions, such as acceptable

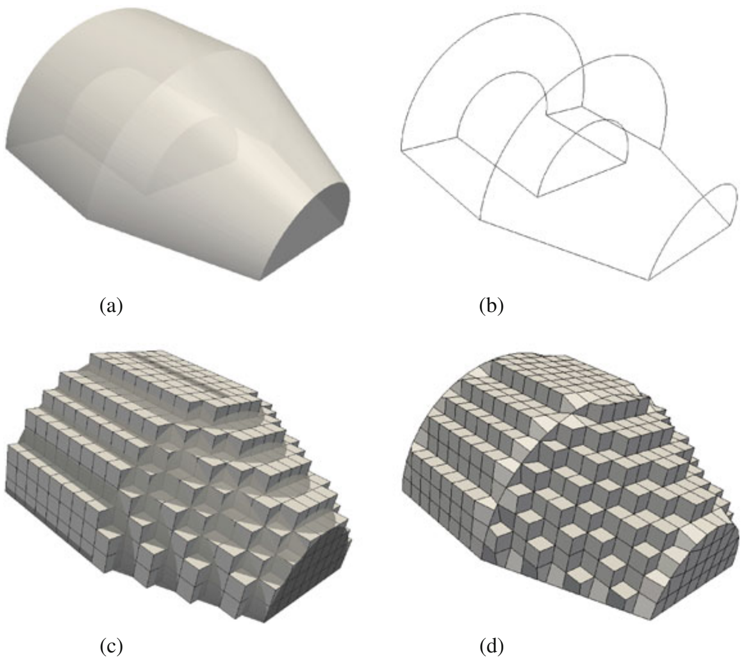

Fig. 2 Hexahedral mesh construction. (**a**) STL-geometry. (**b**) Model's «frame». (**c**) Structured hex-grid (with background STL surface). (**d**) Grid projected to the model's «frame»

Algorithm 1 Orthogonal structured grid generation

```
1: for $z = 1; z < \dim_z / h; z++$ do
2:     for $y = 1; y < \dim_y / h; y++$ do
3:         for $x = 1; x < \dim_x / h; x++$ do
4:             if $sgn_{\mathcal{F}}^{(x,y,z)} + sgn_{\mathcal{F}}^{(x+h,y,z)} + sgn_{\mathcal{F}}^{(x+h,y+h,z)} + sgn_{\mathcal{F}}^{(x,y+h,z)} + sgn_{\mathcal{F}}^{(x,y,z+h)} + sgn_{\mathcal{F}}^{(x+h,y,z+h)} + sgn_{\mathcal{F}}^{(x+h,y+h,z+h)} + sgn_{\mathcal{F}}^{(x,y+h,z+h)} \geqslant \gamma$ then
5:             end if
6:             Create hexagon $c$ from the voxels $\left\{\boldsymbol{v}^{(x+i,y+j,z+k)}\right\}_{i,j,k=0,h}$
7:         end for
8:     end for
9: end for
```

distance of voxels to the surface. For example, the cell c is not constructed if $\exists \boldsymbol{v}' \in \left\{\boldsymbol{v}^{(x+i,y+j,z+k)}\right\}$, $i, j, k = 0, h$ such that $\mathcal{F}\left(\boldsymbol{v}'\right) < \varepsilon_0 < 0$, where ε_0 is some threshold value defined by the user.

The third step is the projection of the orthogonal grid onto the set Λ to provide an approximation of «characteristic» edges and nodes of the model. This procedure is slightly different from the isomorphism technique proposed by Schneiders [10]. In our case, the adaptation is performed before running any smoothing or topological operations with the mesh, and it can be divided into 3 main steps:

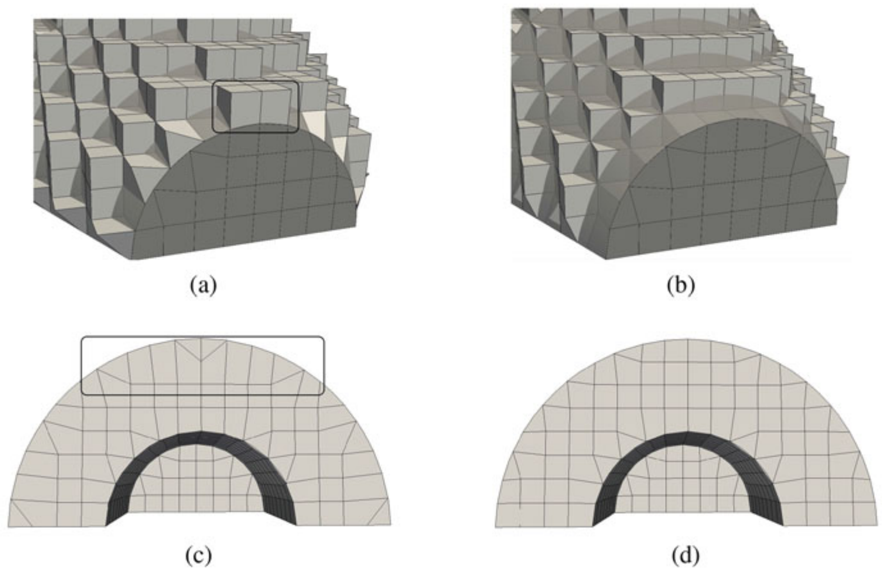

Fig. 3 The choice of the number γ of positive voxels in a cell during structured grid generation: (**a**, **b**) additional hexahedra (in frame) in the case of $\gamma < 4$ in comparison with $\gamma = 4$; (**c**, **d**) lack of uniform cell size (large ones in frame) in the case of $\gamma > 4$ in comparison with $\gamma = 4$

(1) At first on each «characteristic» node η an appropriate grid node $\mathbf{x}$ is projected. Usually, $\mathbf{x}$ is taken as the nearest one to η, but sometimes other rules can be adopted. For example, if two or more nodes have the closest distance to η, the node which is connected to all of them should be chosen.
(2) The next step is the covering «characteristic» edges of the model. At first, all boundary nodes with distance to the «characteristic» edges less than some threshold $\varepsilon_1 \approx 0.1 \times h$ are projected onto them.
(3) The remaining uncovered sections of Λ is handling with sequential nodes projection procedure. It starts from every node $\mathbf{x}$ which is already projected onto Λ. The next candidate $\mathbf{x}'$ is selected from boundary neighbors of $\mathbf{x}$ according to proposed priority criteria which work in the following way:

 - The neighboring nodes, which have two or more boundary inverted quads after projection, are not considered.
 - The nodes with 3 adjacent boundary quads are more preferable than others.
 - Among the nodes with the previous two conditions are fulfilled, the one with the shortest distance to the «characteristic» edge should be taken.

Figure 2d demonstrates the result of the described operation. It should be mentioned that every selected node $\mathbf{x}' \in \mathcal{T}$ is projected to the nearest edge $e \in E_\Lambda$, and this position typically doesn't coincide with vertices of the triangular mesh T.

After the set Λ has been constructed, the remaining boundary nodes are projected to the surface of the model. Here we use two different techniques: the well-known ray-tracing method and a special iterative movement through the volume of SDF data field described in [11]. The first procedure is more accurate, but the latter one has better performance in the case of STL geometries with a high number of triangles. Such STL meshes are often required to describe models with curvilinear surfaces. Each iterative movement consists of 2 steps:

– Check an approximation of node **x** to the iso-surface $I_{\mathcal{F}}(\alpha)$. If the value $\mathcal{F}(\mathbf{x})$ is close enough to α, e.g. $|\mathcal{F}(\mathbf{x}) - \alpha| \approx 10^{-4} \times h$, then it can be assumed that **x** locates on the surface and further iterations are not needed.
– Define new coordinates of node by moving it in a small distance $\mathbf{x}^{+} = \mathbf{x} + \delta\mathbf{n}$, where **n** is the normal of vertex **x** and δ the size of iteration step.

In the case of SDF data format, the boundary of the model is set with zero iso-surface of value $\alpha = 0$.

According to our tests, in most cases, the implementation of these procedures allows to fully describe the model so that each «characteristic» node is assigned to one corresponding mesh node, and each «characteristic» edge is completely covered by mesh edges.

As one can see from Fig. 4a, the projection of remaining nodes to the surface allows to fully described initial STL model with sufficient accuracy.

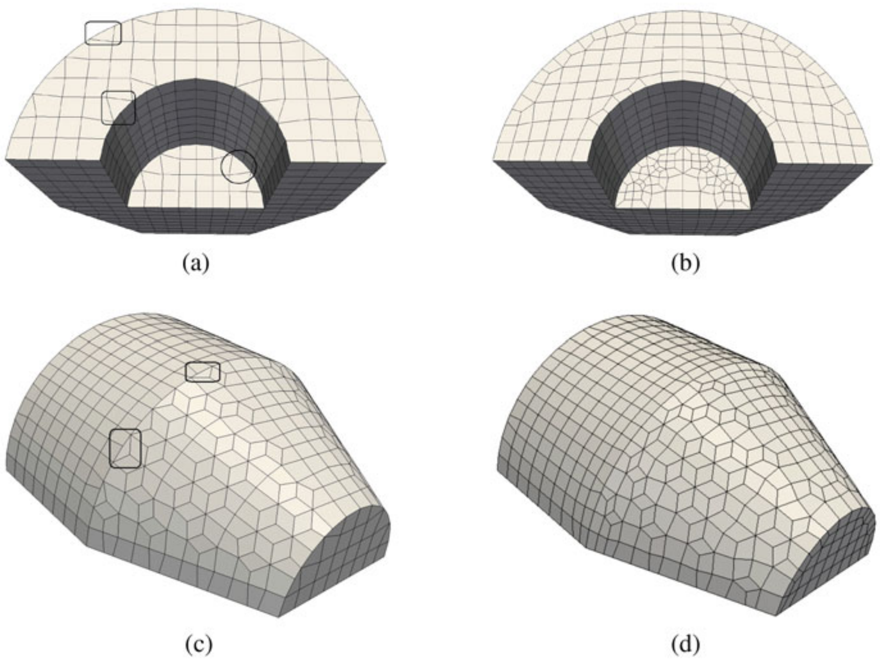

Fig. 4 Removing inverted boundary hexahedra and quads (in frames) with: (**a**, **b**) volume layer insertion and splitting procedures; (**c**, **d**) surface layer insertion along a «characteristic» edge

4 Mesh Quality Improvement

To improve the mesh quality, a Laplacian smoothing procedure is applied. Unfortunately, this operation is not enough to get rid of invalid mesh elements along the model boundary. This is because of hexahedra may have two or three faces to be projected on the same plane, and surface quads may have two edges to be projected on the same sharp line.

To resolve these issues, three standard procedures for grid type meshes are used.

First, a volume layer of hexahedra is inserted around the constructed mesh so that new boundary elements have only one face to be projected on the real geometry (Fig. 4a and b). The shape of boundary quads with angle $\geqslant 180°$ (two are marked with rectangular frames in Fig. 4a) are significantly improved after the volume layer of hexahedra has been inserted (Fig. 4b).

It should be mentioned, that sometimes for better mesh quality it is useful to skip the insertion of volume layer along surfaces which are orthogonal to the Cartesian axes. For instance, for the final mesh displayed in Fig. 4d, bottom, front, and back boundary surfaces stay unchanged, and only for the top curvilinear surface an additional layer of cells was added.

Likewise, a second layer of hexahedra is inserted along the set of «characteristic» edges, if the solid angle alongside them is sufficiently smaller than 180°. Also, another restriction exists, the operation is not allowed if the number of «characteristic» edges adjacent to a «characteristic» point does not equal 2 or 3.

An application of this optimization step is demonstrated in Fig. 4c and d. All hexahedra with flat angles (two are marked in frames) have been removed.

The remaining degenerate elements are removed by a splitting procedure (one is marked with circular frame in Fig. 4c) proposed in [10]. All hexahedra with an inverted face are split into four new ones by specific template (Fig. 4d), thus, the degenerate boundary quad along «characteristic» edge has vanished. To maintain the conformity of the mesh, the neighbor elements are also split up (Fig. 4d).

Finally, to improve the mesh quality, a smoothing procedure is applied. We use different types of Laplacian smoothing weighted by the edges length, quads areas, and volume of hexahedra. It should be mentioned that often a simple Laplacian technique is not enough to get properly shaped cells.

One of the options is to use the optimization-based smoothing method proposed in [2]. In our algorithm, we apply a modification of this method for the case of hexahedral cells. This approach is an iterative one and could improve the value of any hexahedra quality measure q which varies in the interval $[-1; 1]$.

The algorithm works with three quality criteria for hexahedra cells, among which are scaled Jacobian, inverse aspect ratio, and skew metric. Also, we propose a normalized measure for a warping angle Θ of a quadrilateral face $q_\Theta = (\pi - \Theta)/\pi$.

Since the optimization-based technique is time-consuming, it is often applied to the nodes with poor quality cells. In other cases, various types of Laplacian smoothing are used [2].

5 Test Examples

In this section, the results of the algorithm on three types of STL geometry and one volumetric data representation are shown.

All demonstrated hexahedral meshes do not contain inverted cells. The measure of the minimum scaled Jacobian has a value of $Q_J \geqslant 0.2$, which is acceptable for finite element algorithms. This quality metric was chosen as an objective function for the optimization-based smoothing technique. Meshes quality characteristics can be viewed in Table 1.

Figure 5a demonstrates a cut view of the metal sleeve model with a cylindrical inclusion, the result hexahedral mesh contains $N = 15{,}930$ nodes and $C = 13{,}842$ cells. A relatively large value of maximal aspect ratio $Q_a = 33.8$ emerges in the boundary layer cells around inclusion.

Maximal skew metric of cells in the star model is $Q_s = 0.84$. Though, all boundaries of the model are flat, the maximal warp angle of the inner quads of the mesh equals $\Theta_w = 61.1°$. This is because the only scaled Jacobian value was optimized, and other metrics were not taken into account.

Due to the small number of «characterictic» features (only edges and no nodes), the mesh of the spring model has better values of scaled jacobian $Q_J = 0.4$ and aspect ratio $Q_a = 4.2$. The most optimal skew value $Q_s = 0.63$ was obtained for a cube model containing 13 elliptic inclusions.

The human head model (Fig. 1) was generated from a DICOM volume data file. Hounsfield scale values (or the values of extracted isosurfaces) for each material were the following: soft tissue $\alpha = -720$, brain $\alpha = 1$, skull $\alpha = 172$. In addition to scaled Jacobian, a warping angle of quadrangles was also improved. Quality measures of the resulting mesh can also be seen in Table 1.

Table 1 Meshes characteristics

Characteristic	Sleeve	Star	Spring	Cube	Head
Dim	$70 \times 129 \times 129$	$129 \times 129 \times 28$	$86 \times 151 \times 86$	65^3	1029^3
h	4	4	1	1	2
N	15,930	3894	239,757	288,152	1,363,369
C	13,842	2860	195,904	276,003	1,308,341
Θ_w	73.6°	61.1°	54.3°	57.1°	37°
Q_J	0.22	0.29	0.4	0.4	0.2
Q_a	33.8	20.7	4.2	4.9	15.9
Q_s	0.84	0.84	0.8	0.63	0.85

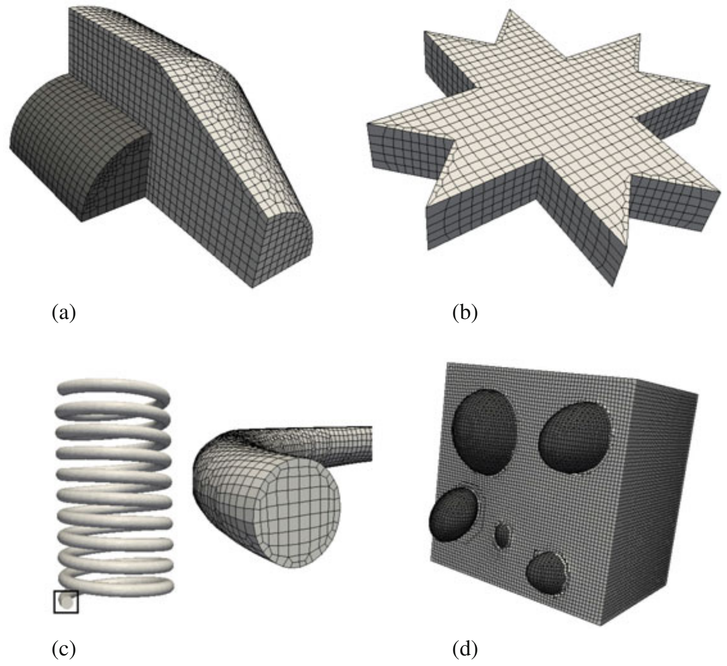

Fig. 5 Unstructured hexahedral mesh examples. (**a**) Sleeve with inclusion. (**b**) Star. (**c**) Spring. (**d**) Cube with inclusions

6 Conclusions

This paper presented a further modification of early developed volume hexahedral mesh generator as applied to the models described by STL geometry.

The main developments of new functionality are the following:

- Obtaining from the original STL geometry a set Λ of «characteristic» edges and nodes, which positions should be saved in the resulting hexahedral mesh $\mathcal{T}$. The set Λ is considered to be a «frame» of the model, which preserves all its sharp angles and faces.
- Construction and mapping the part of boundary nodes of $\mathcal{T}$ onto the set Λ to provide an approximation of boundary contour. The procedure is based on a sequential node projection at every element of Λ. Here is the most suitable nodes are selected according to proposed priority criteria, which evaluate adjacent hexahedra number, quality measures of cells and faces, and the distance to the elements of Λ.
- Removing degenerated cells of $\mathcal{T}$ emerging along «characteristic» edges. The operation includes insertion of an additional surface layer of hexahedra along the edges and splitting inverted cells into four new ones according to the template described in [10].

Implementation of the new procedures allows to generate a uniform mesh accurately describing the original geometry of an arbitrary STL-model.

Generation of multicomponent models for the case of voxel data is carried out by a sequential mesh construction for each isosurface value in descending order.

In the case of a surface representation, the application of such an approach seems difficult, since the SDF file obtained from the STL-geometry contains only one isosurface value $\alpha = 0$. Thus, the SDF file stores information about only one material of the model.

Using one file in the SDF format allows to solve a specific case when the model contains areas with inclusions located entirely inside, or on one level with the orthogonal plane (Fig. 5a and d).

To implement the general case it is necessary to use a set of SDF files, generated for every specific material. Then all orthogonal meshes, obtained from each SDF file, should be assembled in one.

At the moment, an integration of the described technique to the algorithm just like an implementation of adaptive hexahedral mesh generation are questions of further research.

References

1. Awad, M., Rushdi, A., Misarah, A., Mitchell, S., Mahmoud, A., Chandrajit, B., Ebeida, M.: All-hex meshing of multiple-region domains without cleanup. Procedia Eng. **163**, 251–261 (2016). https://doi.org/10.1016/j.proeng.2016.11.055
2. Cannan, S., Tristano, J., Staten, M.: An approach to combined Laplacian and optimization-based Smoothing for triangular, quadrilateral, and quad-dominant meshes. In: 7th International Meshing Roundtable. Dearborn. Michigan, pp. 479–494 (1998)
3. Gao, X., Jakob, W., Tarini, M., Panozzo, D. Robust hex-dominant mesh generation using field-guided polyhedral agglomeration. ACM Trans. Graph. **36**(4), 114 (2017)
4. Gao, X., Shen, H., Panozzo, D.: Feature preserving octree-based hexahedral meshing. Comput. Graph. Forum **38**, 135–149 (2019). https://doi.org/10.1111/cgf.13795
5. Karavaev, A., Kopysov, S.: The method of unstructured hexahedral mesh generation from volumetric data. Comput. Res. Model. **5**, 11–24 (2013). https://doi.org/10.20537/2076-7633-2013-5-1-11-24
6. Ledoux, F., Weill, J.C.: An extension of the reliable Whisker Weaving algorithm. In: Proceedings of the 16th International Meshing Roundtable. Springer, Berlin (2008)
7. Marechal, L.: Advances in octree-based all-hexahedral mesh generation: Handling sharp features. In: Proceedings of the 18th International Meshing Roundtable. Springer, Berlin (2009)
8. Pellerin, J., Johnen, A., Remacle, J.-F.: Identifying combinations of tetrahedra into hexahedra: a vertex based strategy. Procedia Eng. **203**, 2–13 (2017). https://doi.org/10.1016/j.proeng.2017.09.779
9. Ruiz-Girones, E., Roca, X., Sarrate, J.: The receding front method applied to hexahedral mesh generation of exterior domains. Eng. Comput. **28**, 391—408 (2012). https://doi.org/10.1007/s00366-011-0233-y
10. Schneiders, R.: Automatic generation of hexahedral finite element meshes. Comput. Aided Geom. Des. **12**, 693–707 (1995). https://doi.org/10.1016/0167-8396(95)00013-V
11. Zhang, Y., Bajaj, C.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Comput. Methods Appl. Mech. Engrg. **195**(9), 942–960 (2006). https://doi.org/10.1016/j.cma.2005.02.016