STAG: Smart Tools and Applications in Graphics (2025)
M. Comino Trinidad, C. Mancinelli, D. Cabiddu, and D. Giorgi (Editors)

VOLMAP Viewer: a Web-Based Interactive Visual Tool to Explore Volume Maps

Alberto Giacalone*[,1], Dylan Mulas*[,1], Federico Meloni[1], Gianmarco Cherchi[1] and Marco Livesu[2]
*: *joint first authors*

[1] Dept. of Mathematics and Computer Science, University of Cagliari, Italy
[2]CNR IMATI, Genoa, Italy

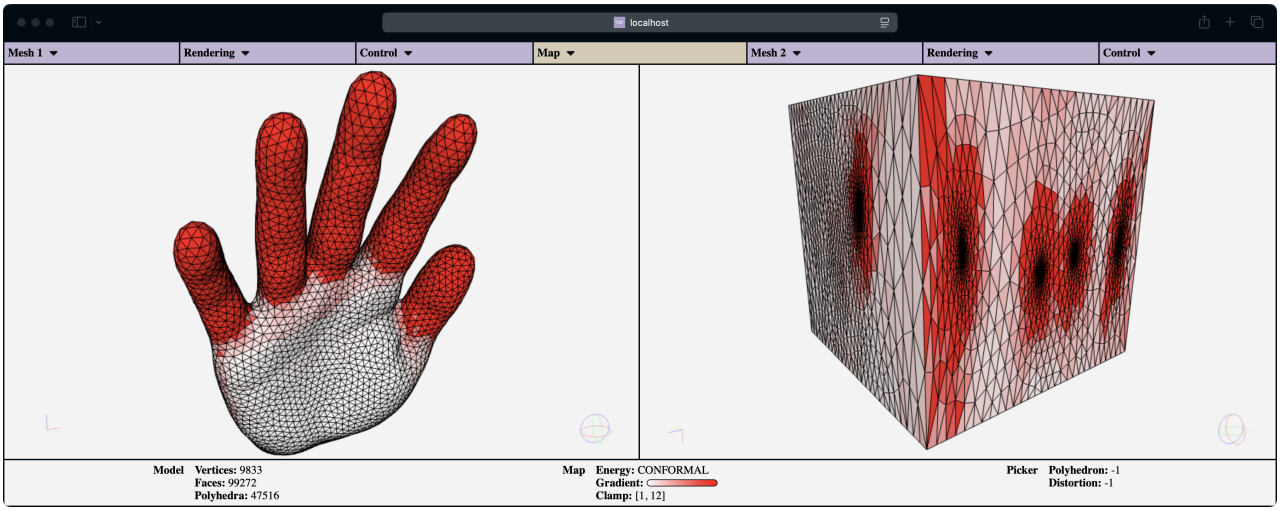


Figure 1: *The VOLMAP Viewer interface enables simultaneous inspection of source and target domains, offering integrated tools for the analysis of their associated map, including distortion-based color coding, element picking, slicing for interior exploration and multiple distortion energies for map quality assessment. In this example, we visualize the conformal distortion of the mapping of a hand onto a cube (data taken from [PLC*21]).*

Abstract

VOLMAP Viewer is a web-based tool for the visual assessment and analysis of maps between tetrahedral meshes. Users can load models directly in the browser and manage them with data structures optimized for interactive visualization. They can display and interact in real-time with both source and target domains, check map validity, and compute map quality according to a variety of popular distortion metrics. Fundamental interaction features such as volume slicing and synchronized picking are also possible: users can select any tetrahedron, highlighting the corresponding element on both domains and inspecting its associated map distortion. Thanks to its portability and interactive performances VOLMAP Viewer complements existing datasets and offline tools, supporting ongoing research on volume mapping algorithms.

CCS Concepts

• ***Computing methodologies*** → ***Graphics systems and interfaces;*** • ***Human-centered computing*** → ***Visualization techniques;***

1. Introduction

Volumetric correspondences between three-dimensional domains enable the faithful transfer of information from one shape to another not only along the boundary but also in the interior. They are frequently used in a variety of applications, such as medicine [ATB*21, DFRVDVM14], digital fabrication [ABC*19, SBR*15, PZM*15], hexahedral meshing [PCS*22], Iso-Geometric Analysis [ZC21, LYLF20] and many others.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

DOI: 10.2312/stag.20251317

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

In most of these cases, the interior of 3D shapes must be mapped to some *canonical* domain that provides a useful structure to solve the problem at hand, such as a cube or a polycube [THCM04]. Generating solid maps to even a trivial shape, such as a cube, has proved to be extremely challenging, and a fair amount of recent research has been devoted to the solution of this critical problem (Section 2). This has triggered the release of tools that are aimed to support this research endeavor, such as VOLMAP [CL23], which provides input data, offline scripts, and C++ code to test and validate mapping algorithms.

This paper presents *VOLMAP Viewer*, a highly portable web-based framework that complements existing tools by assisting volume mapping research from a *visual* standpoint (Figure 1). *VOLMAP Viewer* provides an extremely accessible portal for the interactive assessment of volume maps, which runs entirely in the browser, thus removing the need for dedicated software installation. Its main features include: loading volumetric (tetrahedral) meshes; handling volume inspection operations such as slicing along cutting planes; and computing and displaying popular distortion measures for simplicial maps. The system, as well as its source code, is available as an open source project on the VOLMAP project page (`volmap.github.io`) in the "Viewer" section.

Similarly to other web-based tools such as HexaLab [BTP*19], *VOLMAP Viewer* has been developed using Three.js [CcoT], an open-source JavaScript library that enables the rendering of 3D meshes in a web browser.

In the remainder of this paper, we present in detail all the functionalities offered by *VOLMAP Viewer*, discussing the data structures and core algorithms involved. Particular attention is given to the classes responsible for mesh loading, wrapper data structures, rendering operations, and mapping.

2. Related works

In this section, we briefly present the state of the art for volume mapping algorithms that relate (or could benefit from) our tool, as well as discuss alternative visualization tools that significantly overlap with it.

2.1. Mapping algorithms

Algorithms that rely on the existence of a correspondence (map) between two domains typically require two main properties to be fulfilled: (i) bijectivity, to ensure that no information passed from one domain to the other is duplicated or suppressed; (ii) low geometric distortion, to ensure that local regions maintain comparable shape and size between the two domains. The landscape of methods for volume mapping can be broadly categorized into two families: (i) constructive embedding methods that provide strict guarantees of correctness; (ii) numerical methods that produce a map by minimizing some energy functional.

Constructive methods. Algorithms in this class [NCB23, NCB24, CSZ16, HC23] produce an embedding of a given mesh onto simple base domains such as convex or star-shaped polytopes subject to positional boundary conditions for the outer mesh vertices. Volume mappings to more complex domains can be obtained by coupling them with star-shaped [HBC24] or cuboid [LZS*21] decomposition. In order to guarantee the existence of a solution, embedding algorithms operate both geometrically (assigning vertex coordinates) and topologically (refining the mesh whenever needed). All existing algorithms employ rational numbers to represent vertex coordinates with the necessary accuracy. Partly because of this reason, constructive methods are known to be inefficient and can occasionally run for days before converging to a valid map, even for moderately complex tasks. Memory pressure is also problematic, due to potentially exponential mesh refinement. These features make these methods hardly compatible with general purpose geometry processing pipelines, as discussed in [MCSL24]. It is interesting to notice that solving the mapping problem to convex domains in the volume setting has required the design of entirely novel algorithms, because none of the existing embedding algorithms that solve the equivalent problem in the surface setting are known to extend to volumes [Flo97, SJZP19, Liv24a, Liv24b, FBRCA23, CFLF25]. For some of these methods, there exist numerous counterexamples and theoretical results that ensure that such an extension is not possible [Ale25, CDL95, DVPV03, Liv20a, FPT06]. For the remaining methods, a volumetric extension is perhaps possible, but it has never been discovered yet.

Numerical methods. Methods in this class [OKN21, SFL19, CBC19, ASGS23, DAZ*20, GKK*21, FL16, AL13] operate by minimizing some energy functional that promotes that generation of a valid map, subject to soft or hard positional constraints for the outer mesh vertices, yet possibly subject to additional constraints (e.g., for handling sharp features or positional landmarks). Numerical methods are more general than embedding algorithms, in the sense that they natively handle mappings to domains with arbitrary geometry and topology (genus). Differently from embedding methods, they operate on the vertex coordinates only, without changing the mesh connectivity. The positive aspect of this restriction is that numerical methods are more efficient, both in terms of running times and memory pressure. The negative side is that they may be required to solve a problem that does not admit a valid solution due to instrinsic limitations induced by the topological structure of the input mesh (see Figure 1 in [Liv20b] for a 2D illustration of this limitation). It should be noted that even when a solution exists, numerical methods are not guaranteed to find it. For the class of problems that can be solved also with emebdding methods, they have been shown to fail in a higher number of cases [NCB23, §6.3]. If correctly initialized with a valid (injective) map, some of these methods can reduce geometric distortion while guaranteeing map validity throughout the optimization process, thanks to the use of line search and barrier energies that grow to infinite when one element becomes degenerate [RPPSH17, FL16].

Distortion. Measuring the amount of geometric deformation that each tetrahedron undergoes when moving from one domain to the other is the key metric to assess the quality of a volumetric map. In the literature, numerous alternative distortion metrics exist, each tailored to penalize specific classes of deformations depending on the application context. The most commonly used ones include the *Conformal* energy, which penalizes angle displacements [DMA02];

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

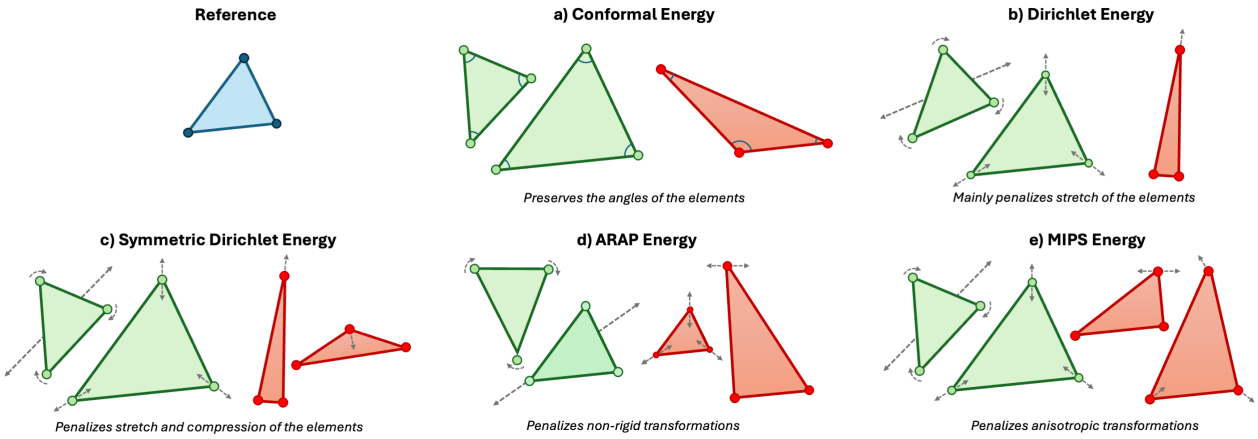


Figure 2: *Comparison of distortion energies on simple element transforms. The reference configuration is shown at left. For each energy (a–e), green examples indicate transformations left invariant (zero or low energy), while red examples illustrate transformations that are penalized (high energy). (a) Conformal: preserves angles. (b) Dirichlet: penalizes stretch. (c) Symmetric Dirichlet: penalizes both stretch and compression. (d) ARAP: penalizes deviations from rigid motion. (e) MIPS: penalizes anisotropic (direction-dependent) scaling.*

the *Dirichlet* energy, which mainly penalizes stretching [Flo03]; the *Symmetric Dirichlet* energy, which simultaneously penalizes stretching and compression [SS15]; *ARAP* (As-Rigid-As-Possible), which discourages non-rigid deformations [SA07]; and *MIPS-3D*, which penalizes anisotropy [AL13]. Figure 2 shows an example of the deformations penalized by the different energies. In general, rigid deformations are never penalized, whereas for non-rigid ones the penalization depends on the specific energy. Maintaining distortion at a minimum is crucial not only to improve the geometric quality of the map but also to ensure numerical robustness. Indeed, when elements approach degeneracy, finite-precision arithmetic may cause interior points of the domain to collapse onto the same discrete location, effectively violating injectivity and compromising the stability of downstream applications. This is one of the reasons why embedding algorithms operate on infinite precision rational numbers. The tool we propose in this paper supports all the aforementioned distortion metrics.

2.2. Visualization

Researchers in geometry processing often support their activity operating on custom-made visualization tools developed ad hoc for their specific experiments, or use third-party software that allows the inspection of only one model at a time [Aya15], without providing functionalities for analyzing the correspondence between two volumetric domains. Furthermore, popular academic visualization tools such as MeshLab [CCC*08] focus on surface meshes only and do not natively support volumetric meshes.

A volumetric map could be easily visualized by writing a few lines of code using many popular C++ libraries that support both tetrahedral meshes and a 3D canvas [S*19, Liv19, JP*18, Lev22]. Despite extremely versatile and open to customization, this solution requires programming capabilities and is not easily portable from one system to the other due to external dependencies and differences among the various operating systems.

This lack of dedicated visualization environments represents a significant limitation, as it makes it harder to interactively explore, debug, and communicate the complex mapping relations that are at the core of volumetric correspondences.

Recently, the HexaLab project [BTP*19] gained popularity in the hexmesh community by proposing a visualization system that runs entirely in the web browser, without requiring technical capabilities to the users for the installation of external software or coding. The tool we propose takes inspiration from the HexaLab project, inheriting its portability and ease of use but adapting its functionalities to the visualization of volumetric correspondences between tetrahedral meshes.

3. The visualization tool

The core of *VOLMAP Viewer* is a web-based application designed for the visualization and analysis of mappings between volumetric meshes composed of tetrahedral elements. Web technologies were intentionally chosen to provide researchers and practitioners with a highly accessible environment for exploring the structure of volumetric correspondences, without the need for installing specialized libraries or relying on substantial computational resources. The entire system is implemented in JavaScript, with the rendering feature developed through the widely used graphics web library Three.js [CcoT]. Although Three.js offers robust support for surface meshes, it does not provide native functionality for volumetric meshes. This limitation motivated the development of dedicated data structures and tailored techniques for the visual inspection of mesh elements that are located in the interior of the model (Section 3.2).

In the following, we present a detailed description of the tool's

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

main components, focusing on data management and rendering strategies, as well as advanced inspection functionalities, including slicing, picking, and map distortion analysis.

3.1. The main window

The main goal of *VOLMAP Viewer* is to provide the core functionalities required for the analysis and inspection of volumetric maps, while maintaining the user experience simple and intuitive. To this end, the tool is structured as a simple browser-based window, split into two adjacent canvases as depicted in Figure 1. Ideally, the left canvas is devoted to the visualization and analysis of the mesh serving as the source domain, while the right canvas offers equivalent capabilities for the target domain. Users can of course load source and target meshes also in the opposite order, but in this case the evaluation of asymmetric distortion energies (e.g., Dirichlet) will be computed with respect to the map that goes from target to source. For symmetric distortion energies (e.g., Symmetric Dirichlet) there is no difference whatsoever.

A control menu positioned at the top of the window enables users to adjust the rendering of the meshes, with standard inspection tools and features specific to map analysis. As it is typically found in standard 3D visualization systems, both for surface and volumetric objects, the tool supports basic operations to control the shading, adjust the wireframe thickness and color, and display the reference axes to indicate the model orientation within the scene.

Crucial features in the tool are the inspection functionalities for internal regions of volumetric models, which are hidden by the opaque outer surface. To address this, the main menu provides access to slicing and picking tools, detailed in Section 3.2. Finally, a dedicated "Map" menu provides access to a set of functionalities developed for analyzing mappings between source and target domains, which constitutes the core contribution of the tool presented in this article (Section 3.4).

To complete the *VOLMAP Viewer* interface, a status bar located at the bottom of the window provides general information about the loaded models, including the number of mesh elements, distortion measures of the analyzed map, and details of individual elements selected through the system's inspection tools.

3.2. Loading and Rendering

VOLMAP Viewer supports tetrahedral meshes in the `.mesh` format [Fre01], which provides both geometric coordinates of vertices and topological connectivity of tetrahedra. Upon loading a model, the system parses the mesh structure and constructs an adjacency map between the model elements. Particularly important is the mapping between faces and tetrahedra, which is crucial for classifying the model's faces as internal or boundary triangles. Boundary faces (i.e., triangles incident to a single tetrahedron) are collected and visualized as an auxiliary surface mesh, enabling a clear view of the object's external (or currently visible) geometry. The visualization of the model at each stage of the inspection operations is subordinate to keeping this auxiliary data structure of visible faces up to date. As we introduced in Section 3, the rendering engine is built on top of the Three.js framework, which does not natively support volumetric meshes. We developed a custom data pipeline to manage tetrahedral structures, ensuring that volumetric connectivity is preserved and that internal faces can be exposed when slicing is applied (Section 3.3). By default, when we load a new model into the system, we render it in a neutral white shading with a black wireframe. This choice emphasizes the model structure and provides a clean and unbiased starting point for analysis, allowing users to become familiar with the structure and the geometry of the input meshes before exploring mapping-related features. Only after the user explicitly selects a mapping energy and activates the visualization of the map distortion, a selected (and customizable) color map replaces the default rendering.

3.3. Volume Inspection

A major challenge in visualizing volumetric meshes is the rendering of internal elements that discretize the volume. While displaying all elements of a surface mesh is sufficient for a reasonable analysis, volumetric meshes require tools that allow users to navigate the interior of the object and inspect the inner tetrahedra through slicing operations. The concept of "*slicing*" an object with cutting planes provides an effective paradigm for examining distortion in volumetric mappings between two objects, enabling analysis not only at the surface but also within internal elements.

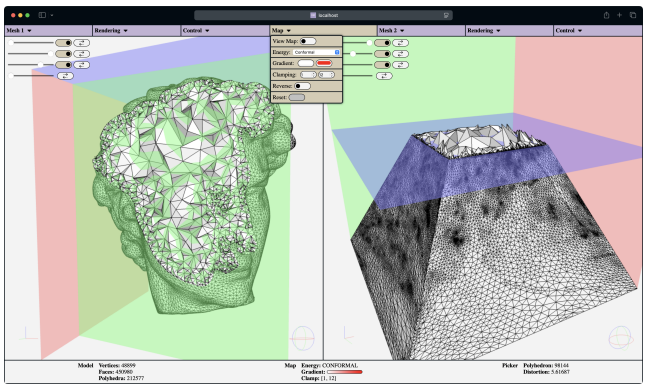


Figure 3: *The possibility to "slice" a volumetric model to inspect its internal geometry, connectivity and distortion is a fundamental task for any volumetric map visualization tool. In VOLMAP Viewer, models can be sliced along the three orthogonal planes aligned with the X, Y, and Z axes, depicted in this figure in red, blue, and green, respectively. In this example, we apply the slicing operation on the* `G1_GM/david` *model from the [MCSL24] dataset.*

In *VOLMAP Viewer*, we provide three independent cutting planes, orthogonal to the X, Y, and Z axes. Each plane can be interactively moved from one end of the model bounding box to the other, according to the represented plane, progressively revealing internal layers of tetrahedra. The planes are optionally rendered as translucent, allowing the user to precisely locate the current slicing position. As we show in Figure 3, to enhance the user experience, each plane is color-coded according to the represented axis: red for the X direction, green for the Z direction, and blue for the Y direction. This convention ensures that users can immediately recognize and distinguish the orientation of the slicing planes during

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

navigation. The three planes can be activated simultaneously, allowing users to carve the mesh into multiple directions at once and obtain clear cross-sectional views of its inner volume. Determining whether a face should be rendered after slicing requires checking its visibility against all three active cutting planes. A triangular face is visualized if and only if it is incident to exactly one visible tetrahedron with respect to the current slicing configuration. Furthermore, for each slicing plane, the orientation can be inverted: in this case, the hidden and visible regions are swapped, effectively flipping the direction from which the model is inspected. This additional flexibility enables the exploration of the volume from different perspectives without requiring manual repositioning of the planes. Notice that, to make these tasks possible in real-time, an efficient implementation of the data structures introduced in Section 3.2 is crucial.

Another important key feature in *VOLMAP Viewer* is the element picking (Figure 4). Picking allows the user to select a tetrahedron in one of the two displayed domains (source or target mesh) and highlight it with a distinct color in both models. When the user select a tetrahedron in a model, the system automatically identifies the corresponding element in the other domain and highlights both simultaneously. This direct correspondence allows the user to verify how a specific region of the domain is mapped into the codomain, and vice versa. In addition to the visual feedback, the bottom information panel displays numerical details about the selected element, including its identifier and the distortion value computed using the selected metric. This combination of complementary highlighting and quantitative reporting links together the visual inspection and the numerical analysis, providing a precise and intuitive way to investigate local properties of the volumetric map.

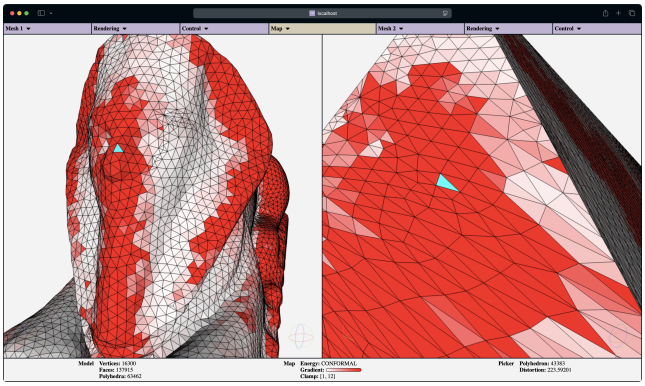


Figure 4: *In analyzing volumetric maps, accessing the information about individual elements within the volume is essential. For this reason,* VOLMAP Viewer *provides the "picking" feature, which allows users to highlight and inspect an element in one domain and locate its corresponding element in the codomain, and vice versa. In this example, we show the picking feature on the* `G1_GM/sphinx` *model from the [MCSL24] dataset.*

3.4. Map Analysis

The main functionality of *VOLMAP Viewer*, beyond the visualization of volumetric meshes described in the previous sections, lies in facilitating the detailed analysis of mappings between volumetric models simply and intuitively, thereby supporting the development, debugging, and assessment of novel volume mapping algorithms.

To be more precise, VOLMAP is designed to operate on piecewise linear simplicial maps which, as often occurs in computer graphics applications, is encoded in two tetrahedral meshes sharing the same connectivity but different embedding (vertex coordinates). The map is therefore trivially defined between pairs of vertices with same ID in the two meshes, and is linearly extended inside each simplex (tetrahedron) by means of barycentric interpolation, which is known to be uniquely defined for simplices in any dimension [Hor14].

Our tool integrates a set of validation methodologies and visualization tools that make the inspection of mappings and associated distortions straightforward and immediate.

In order to check the validity of a mapping between two models, *VOLMAP Viewer* verifies two conditions: (i) the domain and codomain meshes must share the same combinatorial structure, i.e., they must have the same connectivity between elements, and (ii) all tetrahedra must preserve a coherent orientation after the mapping, meaning that the mapping is guaranteed to be locally bijective across the entire volume.

The first condition (i) triggers a loading error if it is not met, notifying the user that the map's structure is not valid.

The second condition (ii), if not satisfied, generates a warning to warn the user that its map is invalid. The orientation check is performed by evaluating the Jacobian matrix of the mapping for each tetrahedron. The Jacobian encodes the local linear transformation that maps the vertices of a domain element to its corresponding image in the codomain. Its determinant provides a scalar measure that gives information about both the scaling and orientation of the transformation. A strictly positive determinant indicates that the analyzed element is mapped in an orientation-preserving manner, maintaining a positive volume. A valid map should satisfy this condition for every tetrahedron in the model to which it is applied, ideally with the least amount of distortion possible. A zero determinant indicates that a domain element has been mapped onto a zero-volume (collapsed) element in the codomain, whereas a negative determinant reveals that the element's volume has changed sign during the mapping operation (resulting in a flipped element). Both conditions make the resulting map invalid, and they represent the most relevant cases to be visualized during the analysis and debugging of an algorithm.

To assess the quality and validity of the computed map, *VOLMAP Viewer* provides several metrics that, by computing the energies introduced in Paragraph 2.1, assign a quantitative value to the distortion encountered by each tetrahedron during the mapping from domain to codomain.

Our tool offers multiple graphical instruments for analyzing distortion values. Through a dedicated menu in the window top toolbar, the user can select the distortion metric of interest and color both models (domain and codomain) accordingly, using a color scheme that highlights the variation of the metric from its minimum to maximum value. By default, distortion values are mapped onto a white-to-red gradient, where white denotes minimal distortion and red marks highly distorted regions. To prevent single outliers from

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

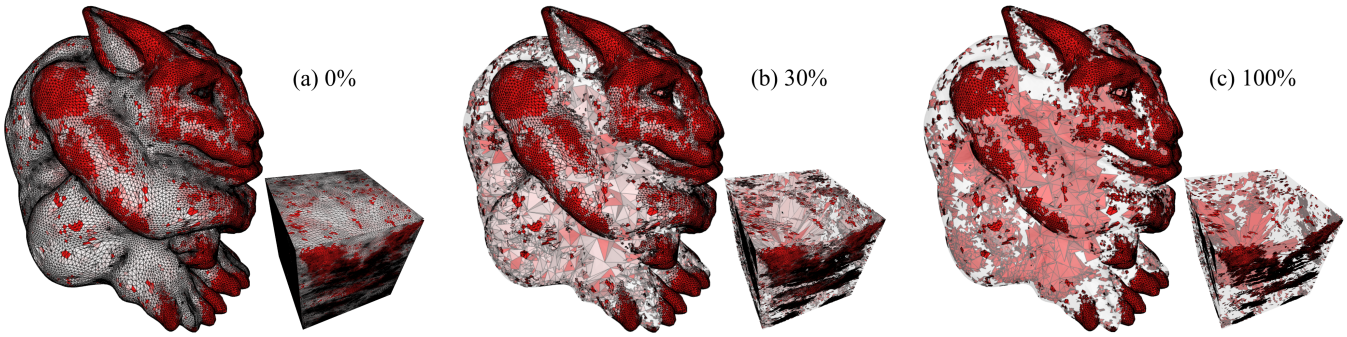


Figure 5: *Visualizing the distortion of both surface and internal elements of a volumetric model is a fundamental task for a tool with the aim to analyze volume maps. To this end, we introduced the "distortion slicing" functionality, which enables users to selectively show or hide the model tetrahedra based on their distortion values. In (a), we display all the mesh elements; in (b), we show the elements with distortion values above the 30% of the distortion range; in (c), we display the elements with the maximum distortion. In this example, we apply the "distortion slicing" to the* `G2_GM/59333_sf` *model from the [MCSL24] dataset.*

dominating the visualization, distortion values are clamped within a predefined range. Each energy is associated with its own default clamping interval, but users can interactively adjust the range with custom values. In addition to the default white-to-red ramp, *VOLMAP Viewer* allows users to define a custom color gradient. Furthermore, since not all energy functions are symmetric, we also allow users to invert the direction of the distortion in the visualization (i.e., from domain to codomain and vice versa), ensuring a more comprehensive analysis of the mapping quality from different perspectives. Table 1 illustrates an example of a model visualized with different metrics available in the system, both using the default values associated with the various distortion energies and with user-defined custom settings.

Since we are dealing with volumetric meshes, the inspection of the mapping quality must be ensured not only for the elements on the mesh surface but also for the elements inside the model volume. In this regard, the slicing functionality already introduced in Section 3.3 is crucial. Specifically, for the analysis of mappings, we introduced an additional slicing feature in our tool based on the values of the metric used to visualize the mapping distortion (an example is shown in Figure 5). Through this "*distortion slicing*" functionality, users can selectively show or hide elements, whether internal or on the surface, that lie above a threshold distortion value defined by the slicer.

This slicing tool, synchronized between the domain and codomain canvases, enables the immediate identification of elements that introduce higher distortion or even invalidate the computed map with collapsed or flipped elements. Such invalid elements are rendered in yellow and are visible when the slicer is positioned on the top value of the range.

3.5. Interactive Performances

The strength of *VOLMAP Viewer* lies in its ability to visualize and inspect mappings between volumetric models, while being usable without the need to install or include external libraries. This motivated the development of a browser-based system built on web

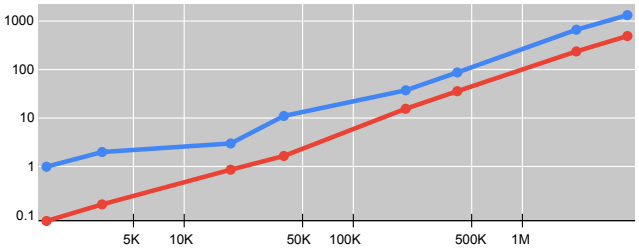


Figure 6: *Log-log plot reporting times (in milliseconds) to complete a planar slicing for meshes with growing number of tetrahedra (horizontal axis), measured on our tool (blue line) and on a popular offline C++ library [Liv19] supporting volume meshes (red line).*

technologies. The use of the Three.js framework, state-of-the-art for 3D model visualization on the web, also required us to complement the volumetric data structure (explicitly implemented for this project) with an additional structure containing only elements that are visible at any given stage of the tool usage. As a consequence, our tool must constantly update this data structure in real time during the interaction between the user and the system. This requirement has a significant impact on the system's real-time performance, which, compared to desktop-based technologies, may lead to increased latency during usage. To provide an estimate of the latency introduced by our tool, we measured the update times of the data structures during a slicing operation across models of different sizes, ranging from 1K to 4M tetrahedra. We then compared the times required for the same task using a similar visualization system developed with a popular C++ offline library [Liv19] (Figure 6), showing that scalability remains linear, with a marginal overhead that does not prevent interactive use for meshes with meshes containing less than a million tetrahedra. Such a range largely covers the scope of existing algorithms for volume mapping.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

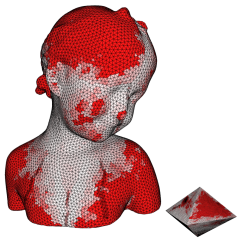
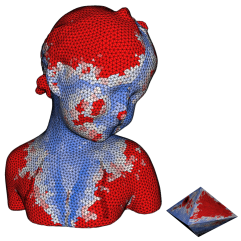
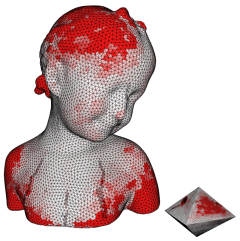
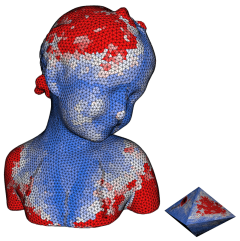
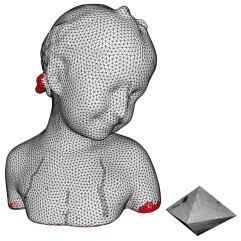
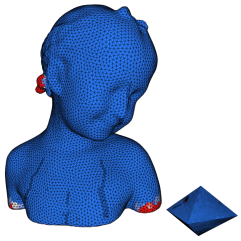
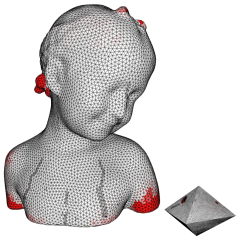
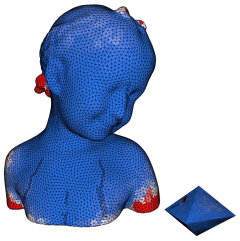
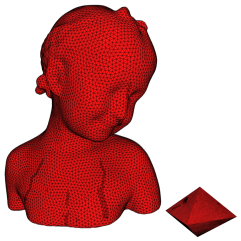
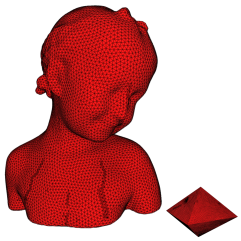
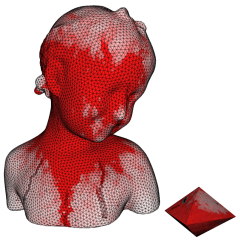
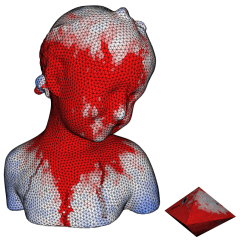
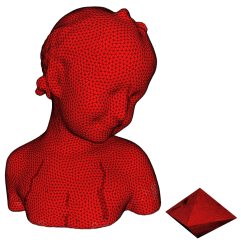
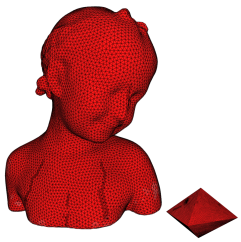
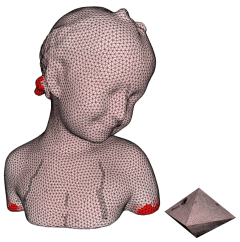
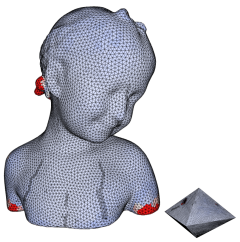
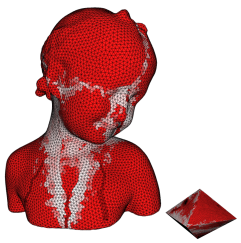
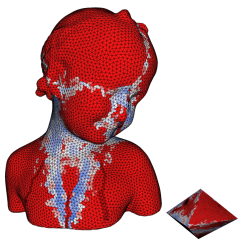
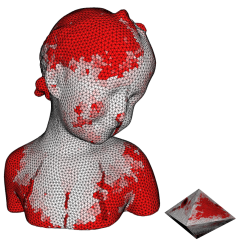
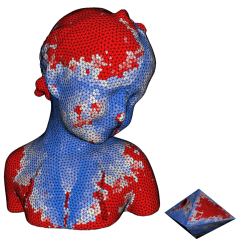
	Default Clamping		Custom Clamping	
	Gradient **White-Red**	Gradient **Blue-Red**	Gradient **White-Red**	Gradient **Blue-Red**
Conformal	 [1, 12]	 [1, 12]	 [1, 30]	 [1, 30]
Dirichlet	 [2, 8]	 [2, 8]	 [0, 1]	 [0, 1]
Sym. Dirichlet	 [3, 12]	 [3, 12]	 [0, 8000]	 [0, 8000]
ARAP	 [0, 2]	 [0, 2]	 [0, 8]	 [0, 8]
Mips 3D	 [1, 12]	 [1, 12]	 [1, 120]	 [1, 120]

Table 1: *The key feature of VOLMAP Viewer is its ability to provide a fast and intuitive check of the validity and quality of a volumetric map, offering, among other tools, several metrics for quantifying the mapping distortion. This table illustrates how the different metrics are visualized, as well as how the color schemes and the minimum and maximum values of the displayed distortion range can be customized by the user (`G1_GM/bimba` model from the [MCSL24] dataset).*

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

4. Conclusions and future work

We presented *VOLMAP Viewer*, a web-based tool designed for the analysis and visualization of volumetric maps.

From an academic perspective, *VOLMAP Viewer* enlarges the set of tools aimed to support research on the generation of volumetric maps in a novel direction, complementing (but also integrating with) previously existing facilities [CL23]. From an application standpoint, the adoption of the Three.js web framework ensures portability and accessibility without requiring dedicated software installation. Similar tools based on the same technology have been successfully released in the past to support research on different geometry processing problems [BTP*19], suggesting that this type of contribution is indeed useful and welcome from the geometry processing community. For this reason, we made *VOLMAP Viewer* available to the geometry processing community as an open source tool at `volmap.github.io/viewer`. The project can be accessed online from any web browser, and can be extended by acting with its source code, which is provided in the repository linked on the project page.

At present, *VOLMAP Viewer* supports tetrahedral meshes stored in the `.mesh` format. This is a widespread format for the exchange of volumetric meshes in academia but is not the only possible format. Future extensions may include support for additional mesh types and file formats, such as hexahedral meshes and `.ovm` or `.vtk` files. More in general, we plan to design a file handling system that easily allows users to incorporate support for their own files. We also plan to integrate support for robust predicates to check map validity, using existing Javascript implementations of the Shewchuk's robust gometric predicates [RS97], but also rational numbers to encode vertex coordinates. As discussed in Section 2, many robust algorithms for volumetric mapping rely on exact arithmetic to guarantee correctness, therefore having the possibility to load results encoded in such a format may be useful, even though for the mere sake of visualization such a precision would be lost anyways due to the unavoidable rounding that occurs when passing data to the video card.

Further enhancements will also focus on expanding the current set of functionalities and improving the user interface. For instance, slicing planes could be released from the restriction of being axis aligned, and the interface could be extended to provide additional numerical information, such as minimum, average, and maximum distortion values, as well as statistics on collapsed or inverted elements. We also plan to introduce new volume inspection facilities, such as the possibility to manually select what elements should be shown/hidden, so as to manually explore the local connectivity in the areas of interest. Similar tools are already part of the volume inspection facilities for both online and offline frameworks [Liv19, BTP*19]. Last but not least, we plan to realize a user study to get feedback from the community, both on the usability of the current functionalities and also to set priorities for the future releseas of the tool.

In summary, *VOLMAP Viewer* can be regarded as a first step towards an increasingly rich and flexible ecosystem of web-based tools for the exploration and analysis of volumetric maps, bridging the gap between theoretical research in geometry processing and practical, interactive inspection of algorithmic results. In this regard, we encourage contributions from researchers and domain experts who wish to propose or implement new features.

Acknowledgments

G. Cherchi, A. Giacalone and D. Mulas acknowledge the support to their research by project "*FIATLUCS - Favorire l'Inclusione e l'Accessibilità per i Trasferimenti Locali Urbani a Cagliari e Sobborghi*" funded by the PNRR RAISE Liguria, Spoke 01. F. Meloni acknowledges financial support through the scholarship "*Tecnologie innovative e abilitanti per l'innovazione della pubblica amministrazione*", founded by PNRR, DM 118/2022. M. Livesu acknowledges the support to his research by project "*RAISE - Robotics and AI for Socio-economic Empowerment*" (ECS00000035) funded by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), PNRR, Mission 4, Component 2, Investment 1.5. M. Livesu also thanks *INdAM (Istituto Nazionale di Alta Matematica "Francesco Severi")* for partially supporting his research.

References

[ABC*19] ANTOLIN P., BUFFA A., COHEN E., DANNENHOFFER J. F., ELBER G., ELGETI S., HAIMES R., RIESENFELD R.: Optimizing micro-tiles in micro-structures as a design paradigm. *Computer-Aided Design 115* (2019), 23–33. 1

[AL13] AIGERMAN N., LIPMAN Y.: Injective and bounded distortion mappings in 3d. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–14. 2, 3

[Ale25] ALEXA M.: Tutte embeddings of tetrahedral meshes. *Discrete & Computational Geometry 73*, 1 (2025), 197–207. 2

[ASGS23] ABULNAGA S. M., STEIN O., GOLLAND P., SOLOMON J.: Symmetric volume maps: Order-invariant volumetric mesh correspondence with free boundary. *ACM Transactions on Graphics 42*, 3 (2023), 1–20. 2

[ATB*21] ABULNAGA S. M., TURK E. A., BESSMELTSEV M., GRANT P. E., SOLOMON J., GOLLAND P.: Volumetric parameterization of the placenta to a flattened template. *IEEE transactions on medical imaging 41*, 4 (2021), 925–936. 1

[Aya15] AYACHIT U.: *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015. 3

[BTP*19] BRACCI M., TARINI M., PIETRONI N., LIVESU M., CIGNONI P.: Hexalab. net: An online viewer for hexahedral meshes. *Computer-Aided Design 110* (2019), 24–36. 2, 3, 8

[CBC19] COHEN D., BEN-CHEN M.: Generalized volumetric foliation from inverted viscous flow. *Computers & Graphics 82* (2019), 152–162. 2

[CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G., ET AL.: Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference* (2008), vol. 2008, Salerno, pp. 129–136. 3

[CcoT] CABELLO R., CONTRIBUTORS OF THREE.JS: Three.js – javascript 3d library. Open source software, MIT License. URL: `https://threejs.org`. 2, 3

[CDL95] CHILAKAMARRI K., DEAN N., LITTMAN M.: Three-dimensional tutte embedding. *Congressus Numerantium* (1995), 129–140. 2

[CFLF25] CHENG Y.-Y., FANG Q., LIU L., FU X.-M.: Divide-and-conquer embedding. In *SIGGRAPH - Conference Papers* (2025), pp. 1–10. 2

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.

[CL23] CHERCHI G., LIVESU M.: Volmap: a large scale benchmark for volume mappings to simple base domains. *Computer Graphics Forum 42*, 5 (2023). doi:10.1111/cgf.14915. 2, 8

[CSZ16] CAMPEN M., SILVA C. T., ZORIN D.: Bijective maps from simplicial foliations. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–15. 2

[DAZ*20] DU X., AIGERMAN N., ZHOU Q., KOVALSKY S. Z., YAN Y., KAUFMAN D. M., JU T.: Lifting simplices to find injectivity. *ACM Trans. Graph. 39*, 4 (2020), 120. 2

[DFRVDVM14] DEPEURSINGE A., FONCUBIERTA-RODRIGUEZ A., VAN DE VILLE D., MÜLLER H.: Three-dimensional solid texture analysis in biomedical imaging: review and opportunities. *Medical image analysis 18*, 1 (2014), 176–196. 1

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 209–218. 2

[DVPV03] DE VERDIERE É. C., POCCHIOLA M., VEGTER G.: Tutte's barycenter method applied to isotopies. *Computational Geometry 26*, 1 (2003), 81–97. 2

[FBRCA23] FINNENDAHL U., BOGIOKAS D., ROBLES CERVANTES P., ALEXA M.: Efficient embeddings in exact arithmetic. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–17. 2

[FL16] FU X.-M., LIU Y.: Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 1–12. 2

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer aided geometric design 14*, 3 (1997), 231–250. 2

[Flo03] FLOATER M.: One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation 72*, 242 (2003), 685–696. 3

[FPT06] FLOATER M. S., PHAM-TRONG V.: Convex combination maps over triangulations, tilings, and tetrahedral meshes. *Advances in Computational Mathematics 25*, 4 (2006), 347–356. 2

[Fre01] FREY P.: *Medit: An interactive mesh visualization software*. PhD thesis, INRIA, 2001. 4

[GKK*21] GARANZHA V., KAPORIN I., KUDRYAVTSEVA L., PROTAIS F., RAY N., SOKOLOV D.: Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–16. 2

[HBC24] HINDERINK S., BRUECKLER H., CAMPEN M.: Bijective volumetric mapping via star decomposition. *ACM Transactions on Graphics (TOG) 43*, 6 (2024), 1–11. 2

[HC23] HINDERINK S., CAMPEN M.: Galaxy maps: Localized foliations for bijective volumetric mapping. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–16. 2

[Hor14] HORMANN K.: Barycentric interpolation. In *Approximation Theory XIV: San Antonio 2013*. Springer, 2014, pp. 197–218. 5

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 3

[Lev22] LEVY B.: Graphitethree, 2022. https://github.com/BrunoLevy/GraphiteThree. 3

[Liv19] LIVESU M.: cinolib: a generic programming header only c++ library for processing polygonal and polyhedral meshes. *Transactions on Computational Science XXXIV* (2019). https://github.com/mlivesu/cinolib/. doi:10.1007/978-3-662-59958-7_4. 3, 6, 8

[Liv20a] LIVESU M.: A Mesh Generation Perspective on Robust Mappings. In *Smart Tools and Applications in Graphics (STAG)* (2020), The Eurographics Association. doi:10.2312/stag.20201234. 2

[Liv20b] LIVESU M.: Mapping surfaces with earcut. *arXiv preprint arXiv:2012.08233* (2020). 2

[Liv24a] LIVESU M.: Advancing front surface mapping. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15026. 2

[Liv24b] LIVESU M.: Stripe embedding: Efficient maps with exact numeric computation. *ACM Transactions on Graphics (TOG) 43*, 6 (2024), 1–14. 2

[LYLF20] LIU H., YANG Y., LIU Y., FU X.-M.: Simultaneous interior and boundary optimization of volumetric domain parameterizations for iga. *Computer Aided Geometric Design 79* (2020), 101853. 1

[LZS*21] LI L., ZHANG P., SMIRNOV D., ABULNAGA S. M., SOLOMON J.: Interactive all-hex meshing via cuboid decomposition. *ACM Transactions on Graphics (TOG) 40*, 6 (2021), 1–17. 2

[MCSL24] MELONI F., CHERCHI G., SCATENI R., LIVESU M.: To what extent are existing volume mapping algorithms practically useful? In *Smart Tools and Applications in Graphics* (2024), The Eurographics Association. doi:10.2312/stag.20241346. 2, 4, 5, 6, 7

[NCB23] NIGOLIAN V. Z., CAMPEN M., BOMMES D.: Expansion cones: A progressive volumetric mapping framework. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–19. 2

[NCB24] NIGOLIAN V. Z., CAMPEN M., BOMMES D.: A progressive embedding approach to bijective tetrahedral maps driven by cluster mesh topology. *ACM Transactions on Graphics (TOG) 43*, 6 (2024), 1–14. 2

[OKN21] OVERBY M., KAUFMAN D., NARAIN R.: Globally injective geometry optimization with non-injective steps. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 111–123. 2

[PCS*22] PIETRONI N., CAMPEN M., SHEFFER A., CHERCHI G., BOMMES D., GAO X., SCATENI R., LEDOUX F., REMACLE J., LIVESU M.: Hex-mesh generation and processing: a survey. *ACM transactions on graphics 42*, 2 (2022), 1–44. 1

[PLC*21] PITZALIS L., LIVESU M., CHERCHI G., GOBBETTI E., SCATENI R.: Generalized adaptive refinement for grid-based hexahedral meshing. *ACM Trans. Graph. 40*, 6 (dec 2021). URL: https://doi.org/10.1145/3478513.3480508, doi:10.1145/3478513.3480508. 1

[PZM*15] PANETTA J., ZHOU Q., MALOMO L., PIETRONI N., CIGNONI P., ZORIN D.: Elastic textures for additive fabrication. *ACM Trans. Graph. 34*, 4 (2015), 135–1. 1

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1. 2

[RS97] RICHARD SHEWCHUK J.: Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry 18*, 3 (1997), 305–363. 8

[S*19] SHARP N., ET AL.: Polyscope, 2019. www.polyscope.run. 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry processing* (2007), vol. 4, pp. 109–116. 3

[SBR*15] SCHUMACHER C., BICKEL B., RYS J., MARSCHNER S., DARAIO C., GROSS M.: Microstructures to control elasticity in 3d printing. *ACM Transactions on Graphics (Tog) 34*, 4 (2015), 1–13. 1

[SFL19] SU J.-P., FU X.-M., LIU L.: Practical foldover-free volumetric mapping construction. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 287–297. 2

[SJZP19] SHEN H., JIANG Z., ZORIN D., PANOZZO D.: Progressive embedding. *ACM Transactions on Graphics 38*, 4 (2019). 2

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–9. 3

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *ACM transactions on graphics (TOG) 23*, 3 (2004), 853–860. 2

[ZC21] ZHENG Y., CHEN F.: Volumetric boundary correspondence for isogeometric analysis based on unbalanced optimal transport. *Computer-Aided Design 140* (2021), 103078. 1

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.