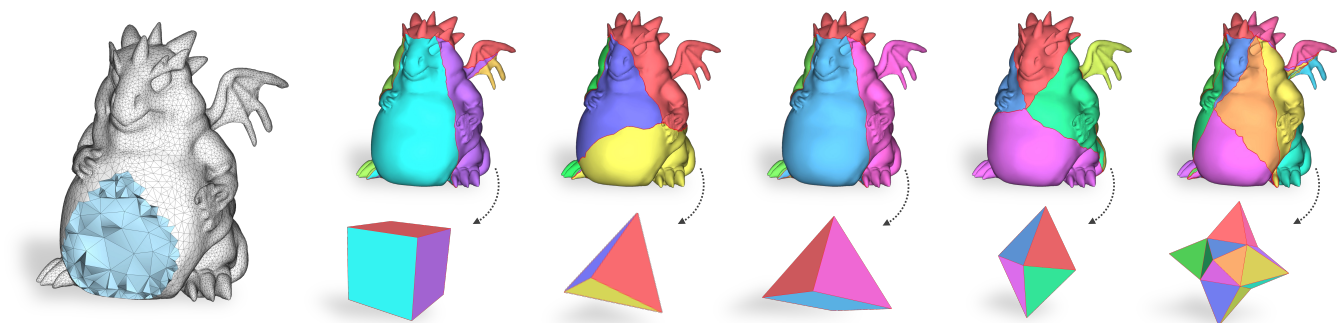# VOLMAP: a Large Scale Benchmark for
Volume Mappings to Simple Base Domains

G. Cherchi [ID]   M. Livesu [ID]
University of Cagliari   CNR IMATI, Genoa

Figure 1: *A tetrahedral mesh (left) and its associated boundary mappings to five alternative base domains supported in VOLMAP (right): cube, tetrahedron, pyramid, octahedron and star. VOLMAP also provides boundary maps to spheres and simple (star-shaped) polycubes. Model:* `39507.mesh` *from group G3.*

Abstract

Correspondences between geometric domains (mappings) are ubiquitous in computer graphics and engineering, both for a variety of downstream applications and as core building blocks for higher level algorithms. In particular, mapping a shape to a convex or star-shaped domain with simple geometry is a fundamental module in existing pipelines for mesh generation, solid texturing, generation of shape correspondences, advanced manufacturing etc. For the case of surfaces, computing such a mapping with guarantees of injectivity is a solved problem. Conversely, robust algorithms for the generation of injective volume mappings to simple polytopes are yet to be found, making this a fundamental open problem in volume mesh processing. VOLMAP is a large scale benchmark aimed to support ongoing research in volume mapping algorithms. The dataset contains 4.7K tetrahedral meshes, whose boundary vertices are mapped to a variety of simple domains, either convex or star-shaped. This data constitutes the input for candidate algorithms, which are then required to position interior vertices in the domain to obtain a volume map. Overall, this yields more than 22K alternative test cases. VOLMAP also comprises tools to process this data, analyze the resulting maps, and extend the dataset with new meshes, boundary maps and base domains. This article provides a brief overview of the field, discussing its importance and the lack of effective techniques. We then introduce both the dataset and its major features. An example of comparative analysis between two existing methods is also present.

1. Introduction

One-to-one correspondences (mappings) between geometric domains are the underlying workhorse of countless downstream applications in graphics, engineering and medicine, where they are used to transfer signals of various kind from one domain to the other. Of particular interest for many of these applications are the mappings to parametric domains with simple topology and geometry, such as convex polygons [FH05], spheres [GGS03], grid

spaces [BLP*13, PCS*22] or simplexes of base domains that are used for cross-parameterization [HPS08] and morphing [Ale02].

These basic mappings are in fact used as atomic building blocks in many higher level pipelines, where it is often required solving numerous instances of these problems, on input data for which very little assumptions can be made (*in the wild*). For this reason, mapping methods to base domains are expected to operate as trusted *black-boxes*, that is, to be both efficient and unconditionally robust.

While for the surface case there exist robust methods that fulfill these requirements [Tut63, Flo97, GGT06, SJZP19, Liv23a], none of these approaches currently extends to 3D, making the fully robust computation of injective volume mappings a prominent open problem for practitioners in graphics and engineering [FSZ*21, NNZ21]. Indeed, the volume mapping problem has been the subject of extensive research in recent years, but satisfactory solutions to this problem are still lacking (Section 2).

In this paper we introduce VOLMAP (`volmap.github.io`): a dataset aimed to support ongoing research on volume mapping to simple base domains. VOLMAP comprises 4.7K input tetrahedral meshes, all endowed with a set of boundary mappings to a variety of alternative simple base domains, such as cubes, tetrahedra, pyramids, octahedra, spheres and, following the capabilities of recent constructive methods for provably injective mapping [Liv23a, NCB, HC], also star-shaped domains. Overall, VOLMAP amounts to more than 22K alternative inputs for algorithms, which are then required to complete the boundary map we provide by positioning interior vertices inside the base domain, generating an injective volume map. In addition to this, we also release various scripts and code to assess the quality of the generated results, compare to alternative methods, or enrich the dataset with additional meshes, boundary maps or base domains.

Connecting to the evaluation criteria listed in the call for dataset papers, VOLMAP is:

- **Novel:** no alternative dataset for the same task was ever released. Large scale datasets of volumetric meshes were already released for the related problem of untangling a tetrahedral mesh containing inverted elements (e.g., [DAZ*20, DKZ*22]). However, since most of these meshes are not convex or star-shaped, they cannot be used for the task that we aim to support;

- **Impactful:** considering the centrality of volume mappings in many tasks, the growing amount of articles published in recent years, and the lack of input data, we expect VOLMAP to have a significant impact in the graphics community and other communities alike. While there cannot be guarantees in this regard, we observe that prior large scale datasets such as Thingi10K [ZJ16] have arguably increased the evaluation standards, becoming a de facto mandatory validation step for diverse algorithms that make claims on scalability or robustness [HZG*18, HSW*20, CLSA20, LPC22, CPAL22, DA21, TNK22, TBFL19, FDBH22, DAZ*20, GSC21]. With VOLMAP, we wish to extend this practice also to volume mesh processing, contributing to the creation of truly robust volume mapping algorithms;

- **Accessible**: VOLMAP data will be hosted at the servers of CNR IMATI. Prominent datasets of 3D shapes in our community have already been hosted at the same institution for many years[†], thus guaranteeing the reliability and permanent accessibility to the data. Considering the large number of meshes and additional information contained in VOLMAP, the dataset is split into smaller thematic chunks that can be downloaded separately. For file formats, all input meshes are in the popular MESH format, and conversion facilities to alternative VTK formats are also provided. MESH and VTK are arguably the two most widely used file formats to exchange volume meshes in our community, and are largely supported by existing academic and commercial tools.

- **Ethical**: data in VOLMAP was mostly created by processing previously existing meshes included in other datasets, performing tetrahedralization and boundary maps as detailed in Sections 4 and 5. Prior to collecting data, we ensured that all the sources we considered did not prevent the processing or redistribution of their models with a restrictive license.

---

2. Field Overview

The goal of this section is to provide the reader with a comprehensive overview of topics related to volume mapping to simple base domains. We first discuss methods for the generation or repairing of injective mappings, emphasizing the lack of robust volume techniques and the inability to extend the existing surface methods to volumes (Section 2.1). Then, we briefly mention relevant volume pipelines that rely on the existence of injective volume mappings (Section 2.2), also discussing related datasets that were released to the community to support research in these areas.

2.1. Existing Mapping Methodologies

For simplicial complexes of dimension 2 (triangle meshes), fully robust methods to map a surface with simple topology to a convex domain have been known since 1963, when Tutte introduced his celebrated embedding [Tut63]. Geometric and topological relaxations that extend the applicability of Tutte [GGT06, WZ14, XCGL11] or that offer superior robustness against limited precision floating point implementations [SJZP19] have also been proposed. However, none of these methods is applicable to simplicial complexes of dimension 3 (tetrahedral meshes), leaving the volume mapping problem largely unsolved.

Tutte 3D. Trivial extensions of the Tutte embedding to three dimensional spaces are notoriously prone to failures. A variety of failure examples have been shown in literature [CDL95, CSZ16, DVPV03], also for trivial meshes containing only four boundary and two internal vertices [FP06, Liv20a]. Chilakamarri and colleagues conjectured that a subclass of graphs for which the Tutte's theorem extends to 3D exists, without formalizing it [CDL95]. Only recently, Alexa proved that 4-connected graphs not having K_6 and $K_{3,3,1}$ minors admit a valid three-dimensional Tutte embedding [Ale23]. However, quoting the author, *"this result has little direct consequence on the practice of using the commonly generated tetrahedral meshes for creating PL mappings: almost all of them have a K_6 minor and, consequently, a convex combination mapping will likely not be an embedding"*. In Section 8, we test Tutte 3D on our benchmark, verifying that in all our experiments the so generated maps contain inverted elements.

Progressive Embeddings. Floating point implementations of the (2D) Tutte embedding may fail to produce an injective map due to numerical cancellation errors. Shen and colleagues extended the seminal idea of progressive meshes [Hop96] to the generation of simplicial mappings, obtaining higher practical robustness. They called their algorithm Progressive Embeddings [SJZP19]. This method operates by first removing all inverted elements from an existing input map through a sequence of edge collapses and then insert back all the previously removed elements by means of vertex splits, ensuring that no vertex insertion changes the orientation of its incident triangles, thus ensuring injectivity. Despite the main ingredients of this algorithm (edge collapse, vertex split) are well defined also for tetrahedral meshes, extending Progressive Embeddings to the generation of volume maps seems overly complex. In fact, differently from triangle meshes, tetrahedral meshes cannot always be collapsed through a sequence of edge collapses, and even deciding whether a given mesh is collapsible is an NP-complete problem [Tan16, MF08, ADGL16]. Barycentric subdivision ensures collapsibility [AB20] but, even if a valid collapsing sequence of a refined mesh is guaranteed to exist, the size of the search space is exponential w.r.t. the number of mesh edges. Failed attempts to heuristically compute a valid collapsing sequence are reported in [Liv20a, Liv20b]. According to [LN21], the chances to get stuck at an incollapsible configuration along a randomly selected collapsing sequence grow exponentially with the number of simplexes in the mesh.

Numerical approaches. Considering the limits of topological approaches, to date, the most reliable methods for the generation of volume maps is through numerical optimization. As detailed in two recent surveys on the topic [FSZ*21, NNZ21], numerical methods compute injective mappings by solving difficult non convex problems. The vast majority of existing approaches solves directly for the *xyz* coordinates of the mesh vertices. However, explicitly imposing injectivity in this formulation yields cubic constraints, making the numerical problem impossible to optimize with any existing solver. A variety of relaxed formulations that promote injectivity without strictly imposing it have been proposed over the years [AL13, KABL14, KABL15, FL16, SFL19, SLS22, ASGS22, NZZ20, OKN21, POK23, DAZ*20, GKK*21], often showcasing remarkable results even on extremely challenging tests. In Section 8, we tested the authors' implementation of [DAZ*20] on a portion of our dataset: in 34% of the cases it fails to produce an injective map. This emphasizes the complexity of the problem we consider, also suggesting that numerical methodologies that do not offer strict theoretical guarantees of correctness are perhaps intrinsically too brittle to reliably compute mappings to simple base domains. A critical aspect of methods in this class is that they often assume that the input mesh connectivity is fixed. As shown in recent research, this assumption hinders the overall robustness of mapping methods, who often need to refine the mesh to open the space of solutions and permit the existence of a valid mapping [Liv23a, NCB, HC]. VOLMAP is designed to test algorithms also on this aspect, providing inputs for which an injective mapping is not guaranteed to exist for a fixed mesh connectivity.

Foliations. Campen and colleagues propose a topological method to construct bijective volume mappings to cubes and spheres [CSZ16] based on simplicial foliations. To date, this is the only published method that is guaranteed to produce a valid map. However, the algorithm applies only to bi-shellable simplicial complexes and, most importantly, the maps it generates are not piecewise linear. In the article, the authors discuss a technique to convert their mappings into a piecewise linear form, but this operation involves huge refinement steps that trigger mesh growths even by orders of magnitude, making this algorithm hardly usable in practical cases. Since its introduction in 2016, we are not aware of any method that internally uses it to compute volume maps. Another foliation algorithm for volumetric mapping to spheres was presented in [CBC19]. However, in this case the approach is numerical, hence it does not provide guarantees of correctness.

2.2. Applications that Enjoy a Volume Map

Volume mappings to simple geometric domains play an important role in a variety of applications. Here we briefly mention a few important ones.

Solid Texturing. Mapping a mesh to the unit cube is used for solid texturing, which is the analog of 2D texturing in UV mapping [PCOS10]. 3D textures are typically used to design the interior material of objects that would break or be sliced, showing their internal structures [TOII08]. This is relevant to model wood, stones, fruits, but also tissues in medical applications [DFRVDVM14]. Solid texturing is also used in advanced manufacturing to map atomic elements of periodic microstructures inside objects [PZM*15, SBR*15, ABC*19], obtaining lightweight pieces with a controlled physical response. Furthermore, cube mappings are also relevant in CAD, where IGA methods exploit them to fit tensor product splines for the numerical resolution of PDEs [ZC21, YLSF21, LYLF20].

Low Distortion Injective Mappings. Mappings to convex domains are a fundamental building block in pipelines that generate low distortion injective maps to generic shapes. These methods operate by first computing a highly distorted but provably injective map to a convex shape. Then, they iteratively reduce distortion by relaxing vertex positions, using line search to prevent the introduction of inverted elements. This technique was pioneered in [SS15] and then extended in many subsequent works [RPPSH17, LYNF18, JSP17, SYLF20, FW22], also for structured meshing [Liv23b]. Due to the lack of robust methods to map a volume to a convex domain, these methods are currently applicable only to surface meshes, where the initialization step is performed using Tutte or similar techniques [Tut63, SJZP19]. For the case of volumes, it has only been validated for tetrahedral mesh improvement (Figure 18 in [RPPSH17]), because this application does not require the generation of the initial valid map.

Shape Correspondences. Algorithms for the computation of shape correspondences or morphing between shapes rely on the existence of an underlying injective map. Prominent methods in this field operate on intermediate base domains for cross parameterization, consisting in a coarse simplicial complex where both shapes are mapped. Each shape is first decomposed into an atlas of charts. Each chart is then assigned to a face of the base domain and

mapped to it using Tutte [Tut63]. Correspondences are therefore constructed as a global (chart-wise) cross-mapping between all the shapes that map to the same base domain. This simple yet effective idea has been extensively used to generate cross-mappings between shapes [LSS*98, GVSS00, PSS01, KLS03, SAPH04], also supporting arbitrary positional constraints [KSG03, SJZP19]. An extension to volumetric cross-mappings using coarse tetrahedral meshes as base domains seems feasible and likely useful, but once again, the lack of robust volume mapping to a tetrahedron prevents the realization of this neat idea. The tetrahedron is one of the base domains we consider in VOLMAP.

Hexmeshing. Not only single cubes, but also mappings to assemblies of cubes are important. In hexahedral mesh generation, these spaces are used to generate a conforming all-hex mesh connectivity that is then projected inside a target shape through an injective map. Both polycube [LVS*13, CLS16, GSZ11, LZS*21, FBL16, GLYL20, FXBH16, PRR*22, DPM*22, MCBC22, HJS*14], framefield [LLX*12, BBC22, NRP11, PBS20, LZC*18, KLF16, SVB17, CC19, RSR*18, SRUL16] and skeleton-based methods [LAPS17, MCK08, LZLW15] operate on grid spaces of this kind. In the last 10 years there has been a lot of research on this topic in the graphics community, but these technologies still struggle to transition to industry because of their lack of robustness, which largely depends from a lack of injectivity of the underlying maps [PCS*22](§8.2). Practitioners in the field have also released volumetric datasets to support their research activities. The HexaLab project [BTP*19] hosts hexahedral meshes that were produced with these and alternative hexmeshing methods. Conversely, Hex Me If You Can [BRK*22] hosts challenging tetrahedral meshes that can be given in input to these methods. None of these datasets are aimed to support research on the more specific problem that we tackle in VOLMAP. It should be noted that, in the general case, the domains used for hexmesh generation are not convex or star-shaped, therefore they fall outside the scope of VOLMAP. Nevertheless, due to their grid structure they can be easily split into (sub) mappings to single cuboids, thus benefitting from advancements in this field.

3. Anatomy of VOLMAP

The VOLMAP dataset is composed of three main ingredients:

- **Meshes:** we collected 4700 tetrahedral meshes with simple topology (genus zero) from various data sources (Section 4). These meshes exhibit a significant variety in resolution (from 1K to more than 1M elements) and a large variety of geometries and features, spanning from CAD to free-form shapes, scanned objects, and artificial models designed for 3D printing and other applications;

- **Boundary Maps:** the boundary of each tetmesh is associated with a set of surface maps to alternative convex or star shaped domains. For most of the models, we provide mappings to five canonical polyhedra, that is: tetrahedron, cube, pyramid, octahedron and star (Fig. 1). For models that were collected from datasets released with mapping methods, such as [PH03, DAZ*20], we also provide surface mappings to

spheres and simple (star-shaped) polycubes;

- **Tools:** in addition to meshes and boundary maps we also release a variety of tools for quality assessment and to support dataset extensions. This includes: C++ code for the computation of inverted elements, mesh distortions, tetrahedralization, and generation of new boundary maps, possibly on brand-new base domains, as well as scripts to batch process the whole (or a selected portion) of the dataset. A tool for the visual inspection of the generated maps and to plot inverted elements and color-coded distortion metrics is also included in the package.

Intended usage. VOLMAP is designed to provide input data for volume mapping algorithms that operate under strict boundary conditions. As discussed in Section 2, this is relevant for a broad variety of applications. Given a tetrahedral mesh and a surface mapping associated to it, candidate algorithms are expected to compute a volume mapping that positions boundary vertices as indicated by the input surface map, solving for the position of internal vertices so as to generate a piecewise linear injective mapping, that is, a mapping where the linear transformation associated to each tetrahedron does not flip its orientation. A natural consequence of the proposed setup is that the mapped mesh precisely interpolates the target domain without possibility to deviate from it. To this end, VOLMAP is mostly focused on the *validity* of the map and not on its geometric fidelity, which is given for granted. Nevertheless, the benchmark could also be used to asses methods that partially relax boundary conditions (e.g. permitting tangential smoothing along the flat faces or the sharp creases of the boundary domains). No tools to measure geometric fidelity are provided.

Considering the number of meshes in the dataset and the boundary mappings associated to them, overall VOLMAP provides 22642 diverse inputs for benchmarking volume mapping algorithms. It should be noted that these mapping tasks are not guaranteed to be well posed, meaning that a piecewise linear injective map of the input meshes may not exist. This is a wanted feature of the dataset, which aims to evaluate the ability of candidate algorithms to employ local mesh refinement to open the space of solutions and provide a valid mapping. As observed in multiple recent works local refinement is an unavoidable step to ensure unconditional robustness, but maintaining mesh growth within practical bounds remains an open unsolved challenge for volume meshing algorithms [CSZ16, HC, NCB].

File structure. Table 1 summmarizes the structure of the dataset, which consists of approximately 10 Gigabytes of compressed files accessible from volmap.github.io. To simplify download operations we have split the dataset into separated chunks, grouped w.r.t. the origin of the input models and the domain onto which the surface was mapped. To give a practical example, the first group in Table 1 (G1) is split into six zip archives: G1.zip contains the 25 tetrahedral meshes contained in this group; then, surface mappings associated to them are organized in five additional zip files, named G1_cube.zip, G1_tet.zip, G1_pyr.zip, G1_octa.zip and G1_star.zip, respectively. For file naming, we tried to organize data in a way that is both trivially processable with a batch script, but also intuitive for visual inspection of the dataset folders. Whenever possible, we preserved the original name of the

Group	Meshes	Cube	Tet	Pyramid	Octa	Star	Sphere	Polycube	Test pairs
G1	25	25	24	25	24	25	-	-	123
G2	67	67	66	66	67	67	-	-	333
G3	1942	1932	1919	1926	1934	1924	-	-	9635
G4	2633	2607	2366	2460	2477	2611	-	-	12521
G5	26	-	-	-	-	-	26	-	26
G6	4	-	-	-	-	-	-	4	4
TOT	4697	4631	4375	4477	4502	4627	26	4	22642

Table 1: *Composition of the VOLMAP dataset, organized in 6 different groups according to the origin of the source meshes. We detail, for each group, how many mappings are present for each geometric domain. For ease of download, each table entry (excluding the rightmost summary column and bottom row) corresponds to a separated zip file that can be downloaded independently from the others. Note that the amount of maps for a given domain does not always correspond to the number of meshes in the same group. This is because in some cases it was impossible to devise a valid injective surface mapping (details in Section 5).*

source mesh, appending the suffix `_tet`, `_cube`, `_pyr`, `_octa` and `_star` to denote the domain type associated to each surface mapping. Meshes in groups G5 and G6 come from existing mapping datasets and contain maps to spheres and polycubes. Also in this case we follow the same convention, using the additional suffix `_sphere` and `_pc` to denote sphere and polycube base domains.

File formats. All tetrahedral meshes in VOLMAP are stored in the MESH file format. This is arguably one of the most widely used file formats for volume meshes in the field, and it is supported by both commercial and academic tools such as Gmsh [GR09], libigl [JP*18], GeoGram [Lev23], CinoLib [Liv19] and Hexa-Lab [BTP*19]. For maximum compatibility, we also provide tools to convert meshes in the VTK file formats (i.e., `.vtu`, `.vtk`). All the meshes in VOLMAP encode tetrahedra implicitly, through the ordered list of their four corners. Specifically, we stick to the following convention: given a tetraheron *abcd*, its vertices are ordered such that the quantity

$$\det \begin{vmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{vmatrix}$$

is strictly positive. Under this convention, the exact orientation predicate `orient3D` introduced by [She97] is strictly positive as well. Indeed, such a predicate is used by our tools to precisely verify the correctness of the generated maps. For the boundary mappings, we encode per vertex boundary conditions in simple yet easy to parse text files, according to the following convention:

```
id0 x0 y0 z0
id1 x1 y1 z1
...
idn xn yn zn
```

i.e., each surface vertex is encoded in a line of text, reporting its vertex id in the tetrahedral mesh and its prescribed coordinates in the base domain. Vertex ids start from zero. Note that this is the same indexing that users would find in any mesh data structure for geometry processing, but it is not the convention used internally by the `.mesh` format, which enumerates vertex indices starting from

1. Once again, VOLMAP provides all the necessary utilities to process data in this format.

4. Data sources

Meshes in VOLMAP have been collected by a variety of alternative sources, sometimes already composed of tetrahedral volume meshes, some other times composed of triangular surface meshes that were originally assembled for other tasks. In this section, we provide details regarding the source data for each mesh group in Table 1, also clarifying the format of the original data and the type of processing that was necessary to import the models into VOLMAP.

G1. This group contains tetrahedral meshes that were originally contained in a dataset released with [FBL16], where they were used for the generation of volumetric maps in the context of polycube mesh construction. The original dataset counts 106 models, from which we extracted 25 meshes with genus zero. Most of these models are freeform and do not contain sharp features. For the most part, these are well known models in the field, that already appeared in numerous articles prior to be released in this specific dataset.

G2. This group contains a subset of 67 models used in [YFL19] for generating surface polycube mappings. As for the group before, the original dataset contains a higher number of shapes, but only those with trivial topology have been incorporated in VOLMAP. Differently from group G1, meshes in G2 were originally triangle meshes, and we transformed them into volumetric meshes using TetGen [Si15] with the `Yq` flags. The class of shapes in this group is the same as G1.

G3. This group contains 1942 meshes from Thingi10K [ZJ16], a large scale dataset that collects thousands of triangle meshes, ranging from toys to abstract and mechanical shapes, originally designed for 3D printing. Most of the meshes in Thingi10K contain a variety of geometric and topological defects, such as open boundaries, non-manifold edges and vertices or self-intersections. For this reason, we considered the *clean* version of the dataset, which was provided by the authors of [HZG*18], and we used Tetgen [Si15] to tetrahedralize them, using the `Yq` flags. Overall, the dataset con-
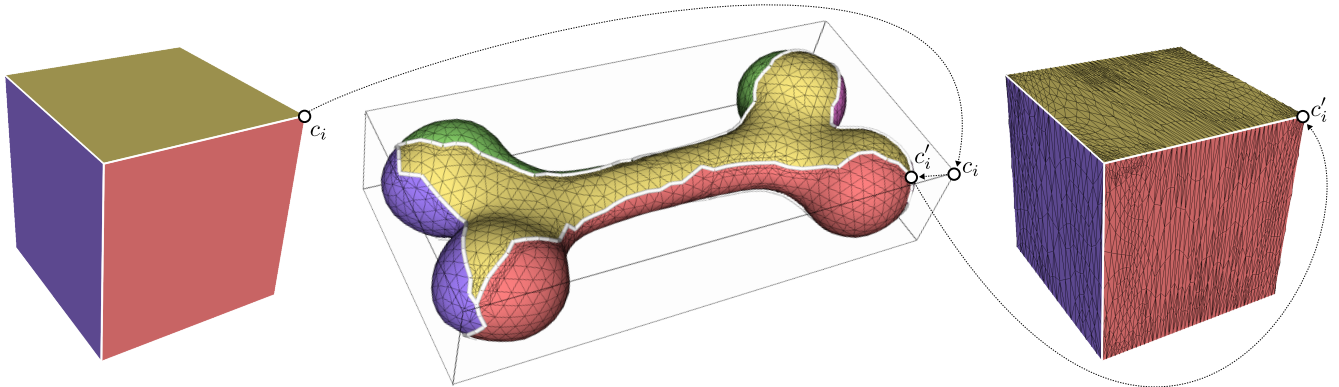
Figure 2: *We map the abstract graph of a base domain (left) by associating its nodes (c_i) to one (or a combination of) the mesh bounding box corners. We then look for the surface vertex (c_i') closest to it (middle). Corners are then connected with Dijkstra, and the Tutte embedding is used to map the surface triangles to each base domain facet, completing the boundary map (right). Model:* `bone.mesh` *from G1 group.*

tains around 9K meshes, but only 1942 of these have genus zero and were included in VOLMAP.

G4. This is the largest group of meshes in VOLMAP. It contains 2633 models taken from the dataset released by the authors of [DAZ*20], and originally distributed by [LYNF18]. This dataset contains shapes of heterogeneous classes (noteworthy is the presence of a big group of humanoids and characters that were not present in previous groups). Since they were originally released to support research in uv mapping and injective surface mappings, the original meshes were mostly topological disks. Luckily, most of these meshes were obtained by starting from a watertight model and cutting it open along a cut graph. For all models generated with this procedure we could therefore reconstruct the original watertight mesh by merging coincident vertices, thus restoring the original topology. In case the so generated mesh had genus zero, we completed the processing by tetrahedralizing it with TetGen [Si15] using the `Yq` flags and including it in VOLMAP.

G5. In this group we collected all mappings to spheres that we could grasp from previously released data. This amounts to 20 volume sphere mappings from [DAZ*20] plus 6 surface sphere mappings computed with [PH03]. For the latter, we transformed the input surface meshes into tetrahedral meshes using TetGen [Si15] with the `Yq` flags.

G6. This group contains 4 models that realize a polycube mapping to a convex or star shaped polycube. These were obtained by processing existing polycube mappings from [YFL19, FBL16] and retaining only polycubes that had a non empty kernel. The kernel (or absence thereof) was checked with [SBS22].

5. Boundary Maps

In this section, we provide details on how the boundary of the tetrahderal meshes in groups G1-G4 have been mapped to the simple domains shown in Fig. 1. Meshes in groups G5 and G6 are not
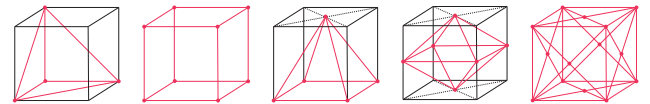
Figure 3: *Base domains (red) are embedded in the input shape by fitting them into the mesh bounding box (black). Specifically, each base domain corner is assigned one or a linear combination of the bounding box corners, and is eventually associated to the mesh vertex that is closest to it.*

considered because they were already endowed with a mapping to a spherical or polycube base domain.
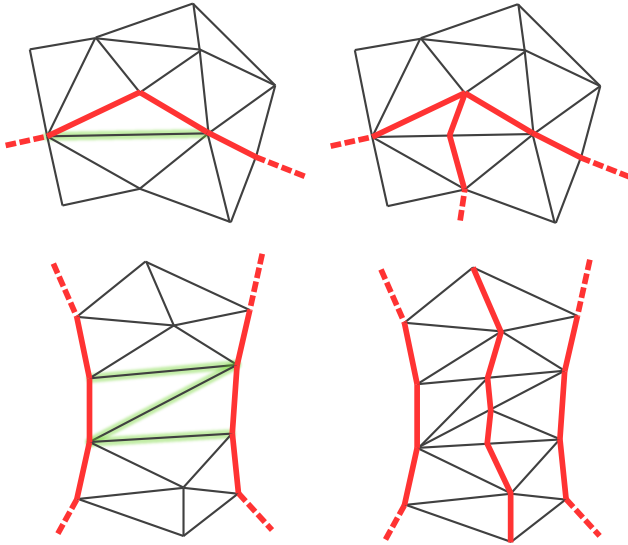
Computing the boundary maps amounts to solving two problems: (i) embed the graph of the base domain on each tetmesh; (ii) map the elements of the chartification induced by such embedding inside each face of the base domain. A pictorial illustration of this pipeline is shown in Fig. 2.

Graph Embedding. Embedding an abstract graph in a mesh is a complex combinatorial problem that may easily produce undesirable poor geometric results [BSK21]. Similarly to existing approaches, we heuristically proceed by first mapping graph nodes to mesh vertices, and then embedding graph arcs in the mesh, defining them as chains of edges connecting the images of their associated graph nodes.

Corners of the base domains are mapped to the mesh by exploiting a correspondence between the topological cube in the left part of Fig. 2 and the eight corners of the mesh bounding box. Once such a correspondence is established, nodes of the base domains are mapped to the mesh by simply locating the vertices that are closest to one (or to a linear combination of) the bounding box corners (Fig. 3). Specifically:
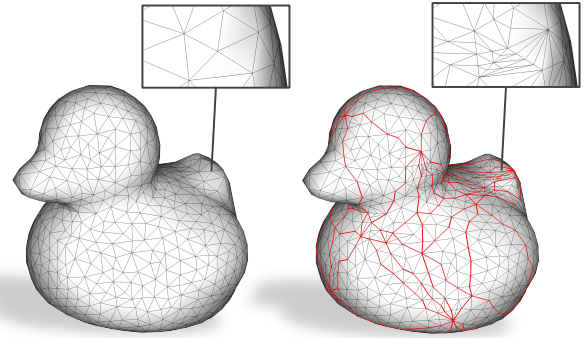
- **cube** nodes are associated to the mesh vertices that are closest to each bounding box corner;

- **tet** nodes are associated to the mesh vertices that are closest to a bounding box corner and to its three adjacent vertices;
- **pyramid** nodes are associated to mesh vertices that are closest to the four bottom corners of the bounding box, plus the vertex closest to the centroid of its upper face;
- **octahedron** nodes are associated to the mesh vertices that are closest to the midpoint of the four vertical edges of the bounding box, plus the points closest to the centroid of its top and bottom faces;
- **star** nodes are associated to the mesh vertices that are closest to all eight corner of the bounding box, plus the centroid of all its faces.

Figure 5: *Example of mesh refinement induced by graph embedding. We show the original model (left) and the embedding of all five base domains in Fig. 1 (right, red edges). Mesh growth was around 5% (from 12601 to 13238 tetrahedra). Model:* `duck.mesh` *from group G1.*

Figure 4: *During graph embedding we always split edges (green shaded) connecting previously embedded graph arcs (in red). We do this both around graph nodes (top) and in narrow passages between adjacent arcs (bottom). This refinement permits to always embed a new graph arc passing in between two previously embedded ones (right column).*

Once all nodes of the base graph have been mapped, we proceed by heuristically inserting one arc at a time, using Dijkstra's algorithm [Dij59] restricted to operate only on the surface edges of the tetrahedral mesh. As shown in [BSK21], the ordering at which arcs are inserted in the mesh may have a dramatic impact on the quality of the embedding, because each insertion creates a non intersection barrier for the subsequent ones, possibly resulting in significant deviations when connecting two nodes of the graph. We avoid excessively distorted results by exploiting local mesh refinement, ensuring that for any pair of embedded arcs there always exists a fully disjoint chain of edges that passes in between them and that can be used for embedding a new arc, if needed (Fig. 4). Note that since our meshes are volumetric, the necessary split operations are performed on the tetrahedral mesh and not only on the surface. Also note that, because of this refinement, tetrahedral meshes in VOLMAP are not identical to the meshes in the source datasets listed in Section 4. In particular, for each model we calculate all the separatrices required for the maps to all five domains, so that the

connectivity of the mesh is modified only once and the same mesh is mapped to all the available domains. Fig. 5 shows an example of mesh structure before and after refinement.

Mapping. Emebdding the graph of each base domain induces a chartification of the mesh surface, which can be trivially computed by exhaustively flooding the surface triangles by starting at a seed and propagating to adjacent triangles without crossing edges associated to a graph arc (Fig. 1). Once a chartification is computed, the boundary mapping is completed by positioning inner vertices of each chart inside the convex faces of the base domain, using the Tutte embedding.

Sanity checks. Despite provably correct, floating point implementations of Tutte 2D may generate vanishing or inverted triangles, preventing the construction of a valid map [SJZP19]. Since surface mappings will be used as strict boundary conditions to initialize a volume map, it is important to ensure that the mapping is fully injective, so as to permit the generation of a valid volume map. To verify the correctness of each result we generate a tetrahedral mesh of each base domain by adding an interior point at the centroid of each polytope and then forming tetrahedra joining such vertex with all boundary triangles. We then use exact orientation predicates [She97] to verify that all the so generated tethrahedra have coherent orientation. Meshes where this check fails are excluded from the dataset. The existence of failures of this kind explains why the number of models in each group reported in Table 1 is slightly higher than the number of actual mappings to each base domain. Missing boundary maps are the ones who failed to be injective due to numerical issues in the computation. Note that while for all the target domains currently considered in the dataset the centroid is contained in the kernel, this property may not hold for alternative more complex star-shaped domains. In this case, the kernel can be explicitly computed – e.g., with [SBS22] – and the kernel centroid used to generate the tetrahedral mesh.
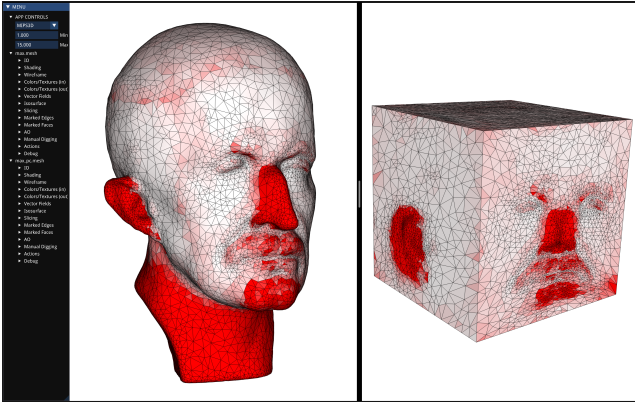
Figure 6: *Snapshot of the visual tool used to inspect a given volumetric mapping, based on [Liv19]. In this example elements are color coded w.r.t. the MIPS 3D geometric distortion, but other distortion metrics can also be chosen. Meshes can be sliced with a cutting plane to inspect the interior of the mapping. Model:* `maxplanck.mesh` *from group G6.*

6. Tools

Besides tetrahedral meshes and their associated boundary mappings, VOLMAP includes a variety of facilities to ease the processing of the dataset, help assess the quality of its results and also extend it with additional models, maps and base domains. In this section, we list and briefly introduce all such tools. All the software facilities we provide are implemented in C++, have minor self-contained external dependencies and can be easily (and automatically) be compiled with CMake on MacOS, Windows and Linux.

Tetmesh checks. This tool ensures that tetrahedral meshes in VOLMAP encode their elements according to the correct convention. Indeed, a tetrahedron can be fully encoded by the list of its four vertices in two alternative ways: given the first three vertices, the fourth one can be positioned either on the positive or the negative half-space defined by the previous three. From the perspective of the correctness of the mapping, either choice is valid. What is important is that the orientation of all mesh elements is *globally* coherent. However, many algorithms make assumptions on these orientations, often expecting a positive one. In case the input mesh follows the opposite convention, this tool flips all its elements, making it compatible with the VOLMAP orientation of choice.

Surface Mapping. This tool implements the graph embedding strategy described in Section 5. The software only depends on CinoLib [Liv19], which is internally used for low level volume mesh processing. Besides the ability to map new tetrahedral meshes to the previously existing base domains, the tool can also be used to define new simple base domains to be included in the dataset. For this latter task, practitioners can provide the description of a novel abstract graph, an embedding of it, and a point within its kernel. The software will then automatically proceed to embed the graph nodes and arcs onto the surface connectivity of the mesh in the pre-

scribed order, also mapping the surface patches of the embedded graph onto the facets of the base domain.

Surface Mapping Check. This tool takes as input a boundary mapping computed with the previous tool and constructs a tetrahedral mesh of its interior to verify that all tets have strictly positive orientation, as detailed in Section 5. This can be used to ensure that additional maps added to VOLMAP fulfill all the necessary correctness requirements.

Metrics. This tool inputs a volume mapping computed with a candidate algorithm and returns information regarding its validity and geometric distortion. For validity, exact orientation predicates [She97] are used to verify that the linear map associated to each tetrahedron has a positive determinant. For geometric distortion, the tool computes the per element distortion energy according to a variety of metrics, such as Conformal [LPRM02], ARAP [LZX*08], MIPS 3D [FLG15], Dirichlet and Symmetric Dirichlet [SS15]. The global distortion energy for each such metric is also returned. This is computed as the normalized sum of per element distortion energies, weighted by the volume of each input tetrahedron. Output mesh size, and in particular its ratio with the input mesh size, can also be regarded as an evaluation metric for methods that employ mesh refinement to ensure the existence of a solution. Such a refinement is in fact aimed to be *minimal*. Finally, in addition to map metrics, we also give the possibility to measure the per element quality of the input or output mesh, measured as the normalized ratio between the incircle and the outcircle, as described in the Verdict Library [SEK*07] (§6.11). In particular, the input mesh quality may be an indicator of the toughness of the mapping task, especially for numerical methods that solve for the vertex coordinates.

Map Visualization. This is the only visual tool provided in VOLMAP. It can be used to visually inspect a volume mapping, plotting its inverted elements and color-coding tetrahdera according to some geometric distortion. A variety of volume inspection tools are supported, such as slicing along axis aligned planes, filtering by per element quality or manual selection of mesh elements to be shown/hidden. All previously mentioned distortion energies are supported and can be selected through the user interface. A screenshot of the application is shown in Fig. 6.

Tetmeshing. This is a simple wrap of Tetgen [Si15] that is included in the package to make it self-contained, and to easily allow practicioners to generate new volume data starting from existing surface meshes.

Scripts. Python scripts to execute the aforementioned tools on large collections of shapes or to launch experiments with a given volume mapping algorithm are also included in the package.

7. How to contribute

VOLMAP is open to new contributions from the community. Practitioners in volume mapping may submit new tracks of the benchmark by simply creating a zip file containing tetrahedral meshes
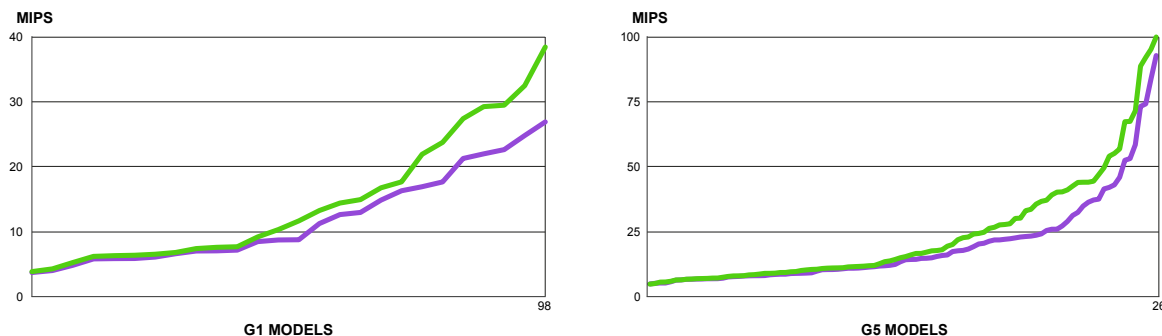
Figure 7: *MIPS3D geometric distortion [FLG15] on volume mappings generated with Tutte 3D (violet) and TLC [DAZ*20] (green) on the groups G1 and G5 of our dataset. Distortion wise methods are comparable, but Tutte 3D introduced inverted elements in all the mappings whereas TLC could make a (barely) injective map in more than half of the cases.*

and boundary conditions according to the formats and structure described in Section 3. Data can be assembled and its correctness be verified either by exploiting the tools described in Section 6 or by using external software. The zip file can then be passed to the authors for verification of correctness and license verification. In case of positive checks, data will then be added to the benchmark, and a new group with a growing progressive number will be assigned to it (e.g., **G7**). In case the submitted surface maps consider a novel target domain, a corresponding suffix for file naming (akin the ones discussed in Section 3) will also be introduced.

8. Example Evaluation

In this section, we report on a simple experiment where we tested Tutte 3D and TLC [DAZ*20] against VOLMAP. We emphasize that these experiments are not meant to rank the two methods, but rather to illustrate how the dataset can be exploited by pratictioners in the field. We also remind the reader that mapping tasks in VOLMAP are not strictly guaranteed to always admit a valid solution, hence since these two methods do not employ local mesh refinement they may be asked to solve an impossible problem. The choice of these two techniques and not others is dictated by the availability of reference code that we could readily use for our experiments. Specifically, for Tutte 3D we used the implementation contained in CinoLib [Liv19], whereas for TLC we considered the author's reference implementation[‡], using the default parameters suggested in the official documentation.

Tutte 3D. We considered mappings to the four canonical domains in Fig. 1 (cube, tet, pyramid and octahedron) for the meshes in group G1 and sphere mappings for the meshes in group G5. Overall, this amounts to 98 experiments for group G1 plus 26 experiments for group G5. For each such experiment we compute the volumetric Tutte embedding by solving a linear system, using the combinatorial Laplacian matrix [XCGL11] and setting (hard) Dirichlet boundary conditions for surface vertices according to the boundary

mappings provided in VOLMAP. Each solve produces an alternative volume map, that we then analyzed as described in the remainder of the section.

TLC. We used the mappings obtained with Tutte 3D to bootstrap the TLC solver, which was then asked to remove all the vanishing or inverted elements contained in the map. TLC requires input data to come in a proprietary format. We exploited the scripts released by the authors to perform this conversion, also using their reference file to set the solver options with default arguments.

Results. The two tests above produce 124 volume mappings each, which we then processed with the tools in Section 6 to extract the number of success and failures and to measure the mapping quality. Not surprisingly, experiments with Tutte 3D confirm that this naive method failed in 100% of the cases, always producing non injective mappings with at least one element having negative volume (computed with exact orient predicates [She97]). Conversely, TLC [DAZ*20] was able to produce a valid map in the 63% of the cases for meshes in G1 (37 failures out of 98) and in the 77% of the cases for meshes in G5 (6 failures out of 26), also exhibiting a slightly lower geometric distortion (Fig. 7). To this end, it should be noted that TLC is not designed to address distortion issues, but rather focuses on the hard problem of removing inverted elements. Overall, this experiment confirms that mappings to simple base domains are extremely complex, and even if TLC is recognized as being one of the most robust numerical methods for injective mapping and mesh untangling, it has collected 44 failures on 124 attempts, suggesting that more research in the field is needed (Section 2.1).

9. Conclusions

We have presented VOLMAP, a novel large scale dataset to support ongoing research on volume mapping algorithms to geometrically and topologically simple base domains. In the first part of the article, we have discussed a variety of applications where this type of mappings are useful, also showing that this is still an extremely challenging problem for which no satisfactory solutions (and dedicated datasets) exist. We have then introduced the VOLMAP components: meshes, boundary mappings, and software tools. Finally,

---

‡ https://github.com/duxingyi-charles/lifting_simplices_to_find_injectivity

we have shown an example of comparative analysis of two volume mapping methods from the state of the art. As predicted in Section 2.1, none of these methods was able to successfully compute all the maps, suggesting that more research in the field is indeed needed to achieve the wanted level of robustness and scalability. We are confident that our contribution will aid practitioners in the field, helping them to assess their algorithms and improve their implementations, finding bugs and unexpected corner cases. Last but not least, we would like to emphasize that VOLMAP is intended to be a *live* dataset, which can grow mainly in two directions: (i) incorporating novel tetrahedral meshes and associated mappings to the known domains; (ii) introducing interesting novel mappings to alternative base domains. Our software tools were designed so as to be modular and to easily support further extensions of the dataset.

Acknowledgements

References

[AB20] ADIPRASITO K. A., BENEDETTI B.: Barycentric subdivisions of convex complexes are collapsible. *Discret. Comput. Geom. 64*, 3 (2020), 608–626. 3

[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. 3

[ADGL16] ATTALI D., DEVILLERS O., GLISSE M., LAZARD S.: Recognizing shrinkable complexes is np-complete. *J. Comput. Geom. 7*, 1 (2016), 430–443. 3

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

[Ale02] ALEXA M.: Recent advances in mesh morphing. In *Computer graphics forum* (2002), vol. 21, Wiley Online Library, pp. 173–198. 1

[Ale23] ALEXA M.: Tutte embeddings of tetrahedral meshes. *Discrete & Computational Geometry* (2023), 1–11. 2

[ASGS22] ABULNAGA S. M., STEIN O., GOLLAND P., SOLOMON J.: Symmetric volume maps: Order-invariant volumetric mesh correspondence with free boundary. *ACM Trans. on Graph. (TOG)* (2022). 3

[BBC22] BRÜCKLER H., BOMMES D., CAMPEN M.: Volume parametrization quantization for hexahedral meshing. *ACM Trans. on Graph. (TOG) 41*, 4 (2022), 1–19. 4

[BLP*13] BOMMES D., LÉVY B., PIETRONI N., PUPPO E., SILVA C., TARINI M., ZORIN D.: Quad-mesh generation and processing: A survey. In *Computer graphics forum* (2013), vol. 32, Wiley Online Library, pp. 51–76. 1

[BRK*22] BEAUFORT P.-A., REBEROL M., KALMYKOV D., LIU H., LEDOUX F., BOMMES D.: Hex me if you can. In *Computer graphics forum* (2022), vol. 41, Wiley Online Library, pp. 125–134. 4

[BSK21] BORN J., SCHMIDT P., KOBBELT L.: Layout embedding via combinatorial optimization. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 277–290. 6, 7

[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. 4, 5

[CBC19] COHEN D., BEN-CHEN M.: Generalized volumetric foliation from inverted viscous flow. *Computers & Graphics 82* (2019), 152–162. 3

[CC19] CORMAN E., CRANE K.: Symmetric moving frames. *ACM Trans. on Graph. (TOG) 38*, 4 (2019), 1–16. 4

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

[CLS16] CHERCHI G., LIVESU M., SCATENI R.: Polycube simplification for coarse layouts of surfaces and volumes. *Computer Graphics Forum 35*, 5 (2016), 11–20. 4

[CLSA20] CHERCHI G., LIVESU M., SCATENI R., ATTENE M.: Fast and robust mesh arrangements using floating-point arithmetic. *ACM Trans. on Graph. (TOG) 39*, 6 (2020). 2

[CPAL22] CHERCHI G., PELLACINI F., ATTENE M., LIVESU M.: Interactive and robust mesh booleans. *ACM Trans. on Graph. (TOG) 41*, 6 (2022), 1–14. 2

[CSZ16] CAMPEN M., SILVA C. T., ZORIN D.: Bijective maps from simplicial foliations. *ACM Trans. Graph. 35*, 4 (2016), 74:1–74:15. 2, 3, 4

[DA21] DIAZZI L., ATTENE M.: Convex polyhedral meshing for robust solid modeling. *ACM Trans. on Graph. (TOG) 40*, 6 (2021), 1–16. 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. on Graph. (TOG) 39*, 4 (2020), 120–1. 2, 3, 4, 6, 9

[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. 3

[Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271. 7

[DKZ*22] DU X., KAUFMAN D. M., ZHOU Q., KOVALSKY S., YAN Y., AIGERMAN N., JU T.: Isometric energies for recovering injectivity in constrained mapping. In *SIGGRAPH Asia 2022 Conference Papers* (2022), pp. 1–9. 3

[DPM*22] DUMERY C., PROTAIS F., MESTRALLET S., BOURCIER C., LEDOUX F.: Evocube: A genetic labelling framework for polycube-maps. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 467–479. 4

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

[FBL16] FU X.-M., BAI C.-Y., LIU Y.: Efficient volumetric polycube-map construction. *Computer Graphics Forum 35*, 7 (2016), 97–106. 4, 5, 6

[FDBH22] FANG X., DESBRUN M., BAO H., HUANG J.: Topocut: fast and robust planar cutting of arbitrary domains. *ACM Trans. Graph. 41*, 4 (2022), 40:1–40:15. 2

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling* (2005), 157–186. 1

[FL16] FU X.-M., LIU Y.: Computing inversion-free mappings by simplex assembly. *ACM Trans. on Graph. (TOG) 35*, 6 (2016), 1–12. 3

[FLG15] FU X.-M., LIU Y., GUO B.: Computing locally injective mappings by advanced mips. *ACM Trans. Graph. 34*, 4 (jul 2015), 8, 9

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

[FP06] FLOATER M. S., PHAM-TRONG V.: Convex combination maps over triangulations, tilings, and tetrahedral meshes. *Adv. Comput. Math. 25*, 4 (2006), 347–356. 2

[FSZ*21] FU X.-M., SU J.-P., ZHAO Z.-Y., FANG Q., YE C., LIU L.: Inversion-free geometric mapping construction: A survey. *Computational Visual Media 7* (2021), 289–318. 2, 3

[FW22] FARGION G., WEBER O.: Globally injective flattening via a reduced harmonic subspace. *ACM Trans. on Graph. (TOG) 41*, 6 (2022), 1–17. 3

[FXBH16] FANG X., XU W., BAO H., HUANG J.: All-hex meshing using closed-form induced polycube. *ACM Trans. on Graph. (TOG) 35*, 4 (2016), 1–9. 4

[GGS03] GOTSMAN C., GU X., SHEFFER A.: Fundamentals of spherical parameterization for 3d meshes. In *ACM SIGGRAPH 2003 Papers*. 2003, pp. 358–363. 1

[GGT06] GORTLER S. J., GOTSMAN C., THURSTON D.: Discrete one-forms on meshes and applications to 3d mesh parameterization. *Comput. Aided Geom. Des. 23*, 2 (2006), 83–112. 2

[GKK*21] GARANZHA V., KAPORIN I., KUDRYAVTSEVA L., PROTAIS F., RAY N., SOKOLOV D.: Foldover-free maps in 50 lines of code. *ACM Trans. on Graph. (TOG) 40*, 4 (2021), 1–16. 3

[GLYL20] GUO H.-X., LIU X., YAN D.-M., LIU Y.: Cut-enhanced polycube-maps for feature-aware all-hex meshing. *ACM Trans. on Graph. (TOG) 39*, 4 (2020), 106–1. 4

[GR09] GEUZAINE C., REMACLE J.-F.: Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering 79*, 11 (2009), 1309–1331. 5

[GSC21] GILLESPIE M., SPRINGBORN B., CRANE K.: Discrete conformal equivalence of polyhedral surfaces. *ACM Trans. Graph. 40*, 4 (2021). 2

[GSZ11] GREGSON J., SHEFFER A., ZHANG E.: All-hex mesh generation via volumetric polycube deformation. In *Computer graphics forum* (2011), vol. 30, Wiley Online Library, pp. 1407–1416. 4

[GVSS00] GUSKOV I., VIDIMČE K., SWELDENS W., SCHRÖDER P.: Normal meshes. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), pp. 95–102. 4

[HC] HINDERINK S., CAMPEN M.: Galaxy maps: Localized foliations for bijective volumetric mapping. *ACM Trans. on Graph. (to appear).* 2, 3, 4

[HJS*14] HUANG J., JIANG T., SHI Z., TONG Y., BAO H., DESBRUN M.: ℓ_1-based construction of polycube maps from complex shapes. *ACM Trans. on Graph. (TOG) 33*, 3 (2014), 1–11. 4

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996* (1996), Fujii J., (Ed.), ACM, pp. 99–108. 3

[HPS08] HORMANN K., POLTHIER K., SHEFFER A.: Mesh parameterization: theory and practice. In *ACM SIGGRAPH ASIA 2008 courses*. 2008, pp. 1–87. 1

[HSW*20] HU Y., SCHNEIDER T., WANG B., ZORIN D., PANOZZO D.: Fast tetrahedral meshing in the wild. *ACM Trans. on Graph. (TOG) 39*, 4 (2020), 117–1. 2

[HZG*18] HU Y., ZHOU Q., GAO X., JACOBSON A., ZORIN D., PANOZZO D.: Tetrahedral meshing in the wild. *ACM Trans. Graph. 37*, 4 (2018), 60–1. 2, 5

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 5

[JSP17] JIANG Z., SCHAEFER S., PANOZZO D.: Simplicial complex augmentation framework for bijective maps. *ACM Trans. on Graph. (TOG) 36*, 6 (2017). 3

[KABL14] KOVALSKY S. Z., AIGERMAN N., BASRI R., LIPMAN Y.: Controlling singular values with semidefinite programming. *ACM Trans. on Graph. (TOG) 33*, 4 (2014), 1–13. 3

[KABL15] KOVALSKY S. Z., AIGERMAN N., BASRI R., LIPMAN Y.: Large-scale bounded distortion mappings. *ACM Trans. on Graph. (TOG) 34*, 6 (2015), 1–10. 3

[KLF16] KOWALSKI N., LEDOUX F., FREY P.: Smoothness driven frame field generation for hexahedral meshing. *Computer-Aided Design 72* (2016), 65 – 77. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation. 4

[KLS03] KHODAKOVSKY A., LITKE N., SCHRÖDER P.: Globally smooth parameterizations with low distortion. *ACM Trans. on Graph. (TOG) 22*, 3 (2003), 350–357. 4

[KSG03] KRAEVOY V., SHEFFER A., GOTSMAN C.: Matchmaker: constructing constrained texture maps. *ACM Trans. on Graph. (TOG) 22*, 3 (2003), 326–333. 4

[LAPS17] LIVESU M., ATTENE M., PATANE G., SPAGNUOLO M.: Explicit cylindrical maps for general tubular shapes. *Computer-Aided Design 90* (2017), 27 – 36. SI:SPM2017. 4

[Lev23] LEVY B.: Geogram, 2023. https://github.com/BrunoLevy/geogram. 5

[Liv19] LIVESU M.: cinolib: a generic programming header only c++ library for processing polygonal and polyhedral meshes. *Transactions on Computational Science XXXIV* (2019), 64–76. 5, 8, 9

[Liv20a] LIVESU M.: A Mesh Generation Perspective on Robust Mappings. In *Smart Tools and Applications in Graphics (STAG)* (2020), The Eurographics Association. 2, 3

[Liv20b] LIVESU M.: Mapping surfaces with earcut. *arXiv preprint arXiv:2012.08233* (2020). 3

[Liv23a] LIVESU M.: Advancing front mapping. *arXiv preprint arXiv:2305.11552* (2023). 2, 3

[Liv23b] LIVESU M.: Towards a robust and portable pipeline for quad meshing: Topological initialization of injective integer grid maps. *Computers & Graphics* (2023). 3

[LLX*12] LI Y., LIU Y., XU W., WANG W., GUO B.: All-hex meshing using singularity-restricted field. *ACM Trans. on Graph. (TOG) 31*, 6 (2012), 1–11. 4

[LN21] LOFANO D., NEWMAN A.: The worst way to collapse a simplex. *Israel Journal of Mathematics 244*, 2 (2021), 625–647. 3

[LPC22] LIVESU M., PITZALIS L., CHERCHI G.: Optimal dual schemes for adaptive grid based hexmeshing. *ACM Trans. on Graph. (TOG) 41*, 2 (2022). 2

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. on Graph. (TOG) 21*, 3 (2002), 362–371. 8

[LSS*98] LEE A. W., SWELDENS W., SCHRÖDER P., COWSAR L., DOBKIN D.: Maps: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), pp. 95–104. 4

[LVS*13] LIVESU M., VINING N., SHEFFER A., GREGSON J., SCATENI R.: Polycut: Monotone graph-cuts for polycube base-complex construction. *ACM Trans. on Graph. (TOG) 32*, 6 (2013). 4

[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. 3

[LYNF18] LIU L., YE C., NI R., FU X.-M.: Progressive parameterizations. *ACM Trans. Graph. 37*, 4 (jul 2018). 3, 6

[LZC*18] LIU H., ZHANG P., CHIEN E., SOLOMON J., BOMMES D.: Singularity-constrained octahedral fields for hexahedral meshing. *ACM Trans. Graph. 37*, 4 (2018), 93–1. 4

[LZLW15] LIU L., ZHANG Y., LIU Y., WANG W.: Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design 58* (2015), 162–172. 4

[LZS*21] LI L., ZHANG P., SMIRNOV D., ABULNAGA S. M., SOLOMON J.: Interactive all-hex meshing via cuboid decomposition. *ACM Trans. on Graph. (TOG) 40*, 6 (2021), 1–17. 4

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 1495–1504. 8

[MCBC22] MANDAD M., CHEN R., BOMMES D., CAMPEN M.: Intrinsic mixed-integer polycubes for hexahedral meshing. *Computer Aided Geometric Design 94* (2022), 102078. 4

[MCK08] MARTIN T., COHEN E., KIRBY M.: Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling* (2008), pp. 269–280. 4

[MF08] MALGOUYRES R., FRANCÉS A. R.: Determining whether a simplicial 3-complex collapses to a 1-complex is np-complete. In *Discrete Geometry for Computer Imagery, 14th IAPR International Conference, DGCI 2008, Lyon, France, April 16-18, 2008. Proceedings* (2008), Coeurjolly D., Sivignon I., Tougne L., Dupont F., (Eds.), vol. 4992 of *Lecture Notes in Computer Science*, Springer, pp. 177–188. 3

[NCB] NIGOLIAN V., CAMPEN M., BOMMES D.: Expansion cones: A progressive volumetric mapping framework. *ACM Trans. on Graph. (to appear)*. 2, 3, 4

[NNZ21] NAITSAT A., NAITZAT G., ZEEVI Y. Y.: On inversion-free mapping and distortion minimization. *Journal of Mathematical Imaging and Vision 63* (2021), 974–1009. 2, 3

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: Cubecover– parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (2011), 1397–1406. 4

[NZZ20] NAITSAT A., ZHU Y., ZEEVI Y. Y.: Adaptive block coordinate descent for distortion optimization. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 360–376. 3

[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. 3

[PBS20] PALMER D., BOMMES D., SOLOMON J.: Algebraic representations for volumetric frame fields. *ACM Trans. Graph. 39*, 2 (Apr. 2020). 4

[PCOS10] PIETRONI N., CIGNONI P., OTADUY M., SCOPIGNO R.: Solid-texture synthesis: a survey. *IEEE Computer graphics and applications 30*, 4 (2010), 74–89. 3

[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 Trans. on Graph. (TOG) 42*, 2 (2022), 1–44. 1, 4

[PH03] PRAUN E., HOPPE H.: Spherical parametrization and remeshing. *ACM Trans. on Graph. (TOG) 22*, 3 (2003), 340–349. 4, 6

[POK23] POYA R., ORTIGOSA R., KIM T.: Geometric optimisation via spectral shifting. *ACM Trans. on Graph. (TOG)* (2023). 3

[PRR*22] PROTAIS F., REBEROL M., RAY N., CORMAN E., LEDOUX F., SOKOLOV D.: Robust quantization for polycube maps. *Computer-Aided Design 150* (2022), 103321. 4

[PSS01] PRAUN E., SWELDENS W., SCHRÖDER P.: Consistent mesh parameterizations. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 179–184. 4

[PZM*15] PANETTA J., ZHOU Q., MALOMO L., PIETRONI N., CIGNONI P., ZORIN D.: Elastic textures for additive fabrication. *ACM Trans. on Graph. (TOG) 34*, 4 (2015), 1–12. 3

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. on Graph. (TOG) 36*, 4 (2017), 1. 3

[RSR*18] RAY N., SOKOLOV D., REBEROL M., LEDOUX F., LÉVY B.: Hex-dominant meshing: mind the gap! *Computer-Aided Design 102* (2018), 94–103. 4

[SAPH04] SCHREINER J., ASIRVATHAM A., PRAUN E., HOPPE H.: Inter-surface mapping. In *ACM SIGGRAPH 2004 Papers*. 2004, pp. 870–877. 4

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

[SBS22] SORGENTE T., BIASOTTI S., SPAGNUOLO M.: Polyhedron kernel computation using a geometric approach. *Computers & Graphics 105* (2022), 94–104. 6, 7

[SEK*07] STIMPSON C., ERNST C., KNUPP P., PÉBAY P., THOMPSON D.: The verdict library reference manual. *Sandia National Laboratories Technical Report 9*, 6 (2007). 8

[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. 3

[She97] SHEWCHUK J. R.: Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry 18* (1997), 305–363. 5, 7, 8, 9

[Si15] SI H.: Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw. 41*, 2 (feb 2015). 5, 6, 8

[SJZP19] SHEN H., JIANG Z., ZORIN D., PANOZZO D.: Progressive embedding. *ACM Trans. on Graph. (TOG) 38*, 4 (2019). 2, 3, 4, 7

[SLS22] STEIN O., LI J., SOLOMON J.: A splitting scheme for flip-free distortion energies. *SIAM Journal on Imaging Sciences 15*, 2 (2022), 925–959. 3

[SRUL16] SOKOLOV D., RAY N., UNTEREINER L., LÉVY B.: Hexahedral-dominant meshing. *ACM Trans. on Graph. (TOG) 35*, 5 (2016), 1–23. 4

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Trans. on Graph. (TOG) 34*, 4 (2015), 1–9. 3, 8

[SVB17] SOLOMON J., VAXMAN A., BOMMES D.: Boundary element octahedral fields in volumes. *ACM Trans. Graph. 36*, 4 (May 2017). 4

[SYLF20] SU J.-P., YE C., LIU L., FU X.-M.: Efficient bijective parameterizations. *ACM Trans. on Graph. (TOG) 39*, 4 (2020), 111–1. 3

[Tan16] TANCER M.: Recognition of collapsible complexes is np-complete. *Discret. Comput. Geom. 55*, 1 (2016), 21–38. 3

[TBFL19] TAO M., BATTY C., FIUME E., LEVIN D. I. W.: Mandoline: robust cut-cell generation for arbitrary triangle meshes. *ACM Trans. Graph. 38*, 6 (2019), 179:1–179:17. 2

[TNK22] TRETTNER P., NEHRING-WIRXEL J., KOBBELT L.: EMBER: exact mesh booleans via efficient & robust local arrangements. *ACM Trans. Graph. 41*, 4 (2022), 39:1–39:15. 2

[TOII08] TAKAYAMA K., OKABE M., IJIRI T., IGARASHI T.: Lapped solid textures: filling a model with anisotropic textures. In *ACM SIGGRAPH 2008 papers*. 2008, pp. 1–9. 3

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society 3*, 1 (1963), 743–767. 2, 3, 4

[WZ14] WEBER O., ZORIN D.: Locally injective parametrization with arbitrary fixed boundaries. *ACM Trans. Graph. 33*, 4 (2014), 75:1–75:12. 2

[XCGL11] XU Y., CHEN R., GOTSMAN C., LIU L.: Embedding a triangular graph within a given boundary. *Computer Aided Geometric Design 28*, 6 (2011), 349–356. 2, 9

[YFL19] YANG Y., FU X.-M., LIU L.: Computing surface polycube-maps by constrained voxelization. *Computer Graphics Forum 38*, 7 (2019), 299–309. 5, 6

[YLSF21] YUAN G.-J., LIU H., SU J.-P., FU X.-M.: Computing planar and volumetric b-spline parameterizations for iga by robust mapping fitting. *Computer Aided Geometric Design 86* (2021), 101968. 3

[ZC21] ZHENG Y., CHEN F.: Volumetric boundary correspondence for isogeometric analysis based on unbalanced optimal transport. *Computer-Aided Design 140* (2021), 103078. 3

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10k: A dataset of 10,000 3d-printing models, 2016. 2, 5