STAG: Smart Tools and Applications in Graphics (2024)
A. Caputo, V. Garro, U. Castellani, and A. Giachetti (Editors)

To What Extent Are Existing Volume Mapping Algorithms Practically Useful?

Federico Meloni[1], Gianmarco Cherchi[1], Riccardo Scateni[1] and Marco Livesu[2]

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

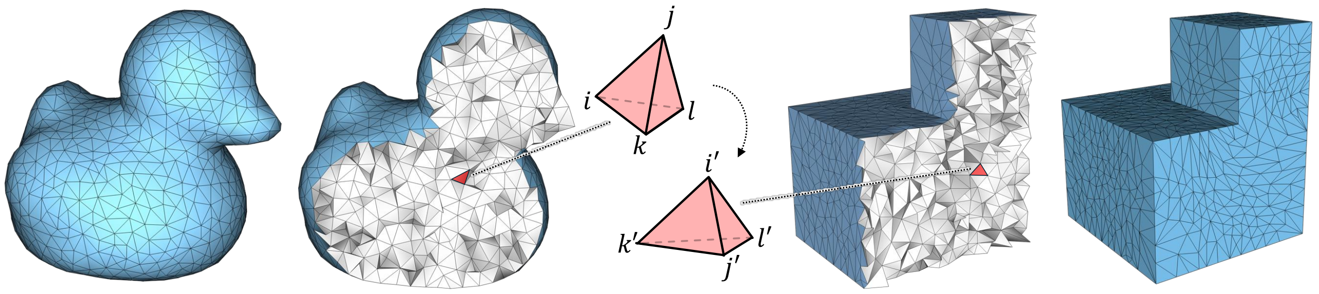

Figure 1: *Mapping a tetrahedral mesh into a new domain is a crucial task in several geometry processing applications. In this example, we consider the model* `duck.mesh` *(left) in the G6 group of the VOLMAP dataset [CL23] and the provided boundary constraints, which map its outer surface onto the surface of a polycube (blue surface on the right). We then compute the volumetric mapping of the interior vertices with [HC23] (one of the two algorithms we analyzed in this work). In the middle, we highlight how a single tetrahedron is deformed during the mapping pipeline.*

Abstract

Mappings between geometric domains play a crucial role in many algorithms in geometry processing and are heavily used in various applications. Despite the significant progress made in recent years, the challenge of reliably mapping two volumes still needs to be solved to an extent that is satisfactory for practical applications. This paper offers a review of provably robust volume mapping algorithms, evaluating their performances in terms of time, memory and ability to generate a correct result both with exact and inexact numerical models. We have chosen and evaluated the two most advanced methods currently available, using a state-of-the-art benchmark designed specifically for this type of analysis. We are sharing both the statistical results and specific volume mappings with the community, which can be utilized by future algorithms for direct comparative analysis. We also provide utilities for reading, writing, and validating volume maps encoded with exact rational coordinates, which is the natural form of output for robust algorithms in this class. All in all, this benchmark offers a neat overview of where do we stand in terms of ability to reliably solve the volume mapping problem, also providing practical data and tools that enable the community to compare future algorithmic developments without the need to re-run existing methods.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Mesh models; Volumetric models; Volumetric Mappings;***

1. Introduction

Computing correspondences between two 3D shapes is a fundamental task in geometry processing. Nearly all applications that require the existence of a map demand that the function that puts into correspondence two domains fulfills two properties: *(i)* it is one-to-one, to reliably transfer a signal from one domain to the other without deleting or creating multiple copies of the same information; *(ii)* it has little geometric distortion, so that local neighborhoods in the two domains maintain a comparable size and shape, enabling a practically useful analysis and processing (Figure 1).

The first requirement is typically a *hard* one, meaning that a map that is not one-to-one is not considered valid and cannot be used.

DOI: 10.2312/stag.20241346

The second is a *soft* requirement, meaning that geometric distortion should be reduced to the minimum but is also expected to be present, to some extent, because limiting to perfectly isometric maps with no distortion is too restrictive and maps of this kind hardly exist in real applications.

Robust methods for the computation of one-to-one maps with low geometric distortion heavily rely on a two-step pipeline in which the hard validity constraint is first addressed without caring about distortion, and then numerical algorithms are bootstrapped with such a valid solution, reducing distortion without introducing collapsed or inverted elements in the domain. Indeed, such an approach has been widely adopted by numerous authors in recent years [RPPSH17, LYNF18, SS15, FW22, JSP17, FL16, SYLF20, YSLF20, WZC*23, Liv24b].

Identifying a valid initial solution is the most critical step for this pipeline. For the surface case, a variety of different algorithms can be used to create an initial one-to-one map by embedding a triangle mesh onto a convex or star-shaped planar domain [Tut63, Flo97, SJZP19, FBRCA23, Liv24a, Liv24b]. Unfortunately, none of these methods is known to extend to the volumetric setting. The problem of embedding a tetrahedral mesh onto a convex or star-shaped polyhedron remains a challenging open problem in the scientific literature, with very few existing alternatives which are, however, severely limited by running times that can stretch up to more than one week of computation and excessive memory consumption (Section 2).

The main goal of this paper is to consider the two most prominent existing methods for the robust computation of one-to-one correspondences between volumes and evaluate their performances in realistic use cases where the runtime and memory requirements cannot stretch indefinitely.

It is essential to underline that this is not a comparison between the two algorithms. It is an attempt to understand to what extent these methods meet the requirements of computer graphics applications. Our analysis confirms that even though both algorithms are formally correct, they exhibit only a limited ability to operate in realistic conditions (Section 4), suggesting that the quest for robust and scalable volume mapping algorithms is still open and demands further research by the geometry processing community.

Algorithms for computing volume mappings – including those considered in this article – often employ critical geometric constructions, such as refinement operations in both the input and the map, that, when implemented in floating point, may easily infringe the bijectivity requirement due to lack of numerical precision. For this reason, rational numbers (i.e., quantities encoded as fractions between integers with arbitrary length) are often used to accurately represent the coordinates of mesh vertices. An additional goal of this article is to share with the geometry processing community the maps we produced using [NCB23] and [HC23] in their exact form. We also provide utilities to load, save and check the validity of maps represented with rational numbers. These utilities, as well as the produced models, have been incorporated into the VOLMAP benchmark [CL23] to facilitate comparisons with new algorithms and foster further research in this field.

2. Related works

We restrict our analysis to methods that guarantee the bijectivity of the map between a volumetric (tetrahedral) mesh and a convex or star-shaped polytope. For a comprehensive discussion on alternative numerical methods covering a broader class of maps but not providing such theoretical guarantees of correctness, we point the reader to two recent surveys [FSZ*21, NNZ21].

Many authors considered the possibility of extending robust surface mapping methods to the volume setting, obtaining only poor results so far. For the volumetric extension of the Tutte embedding [Tut63, Flo97], numerous failure cases have been reported [Liv20b, Liv20a, CDL95, DVPV03, FPT06]. The Tutte embedding was ultimately shown to operate correctly only on a very restricted set of volume graph topologies, which was deemed insufficient to cover practically interesting cases [Ale23]. The Progressive Embedding algorithm [SJZP19] does not extend to volumetric meshes either, because of the difficulty of devising a valid edge collapsing sequence of tetmesh edges during the removal of inverted elements, as discussed in [Liv20b, Liv20a]. More recent methods such as E^3A [FBRCA23], Advancing Front Mapping [Liv24a] and Stripe Embedding [Liv24b] may potentially be extended to the volumetric setting, but no results have been published so far.

To the best of our knowledge only the following three algorithms can compute a provably bijective map between a tetrahedral mesh and a convex or star-shaped polytope.

Simplicial Foliations [CSZ16] was the first algorithm to solve the volume mapping problem for convex domains. The practical utility of this method is hindered by a number of limitations, regarding inefficiency, excessive mesh refinement (by orders of magnitude) to produce a valid piece-wise linear map, and also the impossibility of precisely control the mapping of the outer surface. In fact, users can only choose between radial foliations, for sphere mapping, and parallel foliations, for cube mapping, but not prescribe alternative boundary conditions. This is a major limitation for block decomposition methods that build a complex map by splitting the domain into a number of simpler cuboids welded at shared interfaces, e.g. for hexahedral meshing [BC23, PCS*23]. To our knowledge, a reference implementation of Simplicial Foliations was never released, and the algorithm was never used to solve concrete mapping problems.

The foliation idea was pushed further by the Galaxy Maps algorithm [HC23], which relies on the same concepts but applies them locally, starting from a non-bijective map (e.g., generated with Tutte) and then operating only on small star-shaped clusters containing inverted elements. The method also extended the class of supported domains, from cubes and spheres to any star-shaped polyhedron, fully supporting per vertex boundary conditions. According to the authors, Galaxy Maps gained huge speedups w.r.t. Simplicial Foliations thanks to its local approach, sometimes even by multiple orders of magnitude. However, running times still remain prohibitive and can stretch to even more than 10 days of computation in certain cases (see Figure 18 in their article). Also, in terms of refinement, Galaxy Maps may increase mesh size by more than two orders of magnitude. Despite these limitations, Galaxy Maps has been recently used to compute volume maps in the context of hexmesh generation [BC23], where it was used as a fully

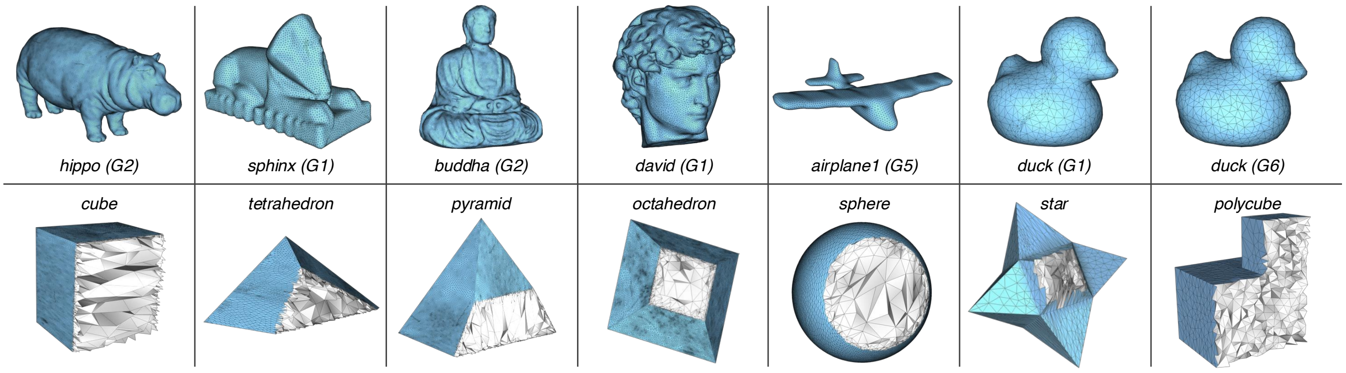

Figure 2: *We tested the two analyzed algorithms [NCB23, HC23] in a subset of models from the VOLMAP [CL23] dataset, which provides thousands of volumetric meshes whose surface is mapped onto the surface of different domains. In this figure, we show some examples of models whose volumetric mapping on the domains of VOLMAP has been computed with [HC23]. Since we share with the community all the produced maps, we report the models' names and the groups they belong to in order to facilitate their identification among the data this work provides.*

robust remedy strategy in cases a non-robust numerical method ([GKK*21]) did not produce a bijective map.

The last existing algorithm for robust volume mapping is Expansion Cones [NCB23], based on the so-called "Shrink and Expand" paradigm. The algorithm first collapses all internal mesh vertices at one location (shrink), generating a mesh in which all internal cells are completely collapsed. Then, in the expansion phase, inner vertices are relocated so as to inflate collapsed cells. Vertex relocation is operated to ensure that all elements obtain a globally coherent orientation, generating a bijective map. To ensure the existence of a valid configuration, the inflation phase may trigger mesh refinement. Similarly to Galaxy Maps, also Expansion Cones is characterized by long running times and excessive memory pressure due to mesh refinement, with mesh growths that can exceed 1K times the initial input size. The authors set a 12h maximum time limit on their experiments, which was reached in 23.38% of cases tested in their article.

It should be noted that all three methods discussed here often require performing on demand refinement in both the input and the map, achieving unconditional numerical robustness by using numeric types and geometric constructions with arbitrary precision. This is a common procedure in critical tasks such as mesh generation and geometric modeling [CLSA20, CPAL22, Lév24, ZGZJ16] and is known to decrease running times by orders of magnitude. In the context of mapping algorithms, the limits of the floating point system were not considered until recently [SJZP19]. The use of fully robust numeric types has been then explored [Liv24a, FBRCA23], observing similar degradation of performances when switching to multi-precision floating points, with one exception [Liv24b].

All in all, existing robust methods are mostly proof-of-concepts that have been validated on limited datasets using high-end machines with almost unlimited hardware resources. In the next session, we will explore the capabilities of these methods on a dedicated dataset that was never considered before [CL23], also putting tight bounds on the hardware capabilities.

3. Setup and contribution

As anticipated in the previous section, we focus our analysis on the two most recent state-of-the-art algorithms: Expansion Cones [NCB23] and Galaxy Maps [HC23]. For brevity, in the remainder of the article, we will refer to them as *EC* and *GM*, respectively.

All experiments have been executed by considering the reference implementation released by the authors, which we tested on alternative input data that was not considered in the original articles. Specifically, we bootstrapped GM and EC with the data provided by the VOLMAP benchmark [CL23], which comprises both input tetrahedral meshes and boundary conditions that map their surface vertices to a variety of alternative target domains, both convex (cubes, tetrahedra, pyramid, octahedra, and sphere) and star-shaped (stars, polycubes).

Specifically, we considered the groups G1 (25 models) and G2 (67 models) with boundary constraints on the cube, tetrahedron, pyramid, octahedron, and star domains, G5 (26 models) with boundary constraints on a sphere and G6 (4 models) on generic polycubes. A mosaic with examples of models mapped into the VOLMAP domains is displayed in Figure 2. Overall, our testing data counts 486 pairs of input models and corresponding boundary constraints.

For ease of further analysis and comparisons with new methods, we release the maps computed with the two algorithms in VOLMAP [CL23], using the following syntax:

- `model_input.mesh`: contains both the connectivity of the input mesh and its original vertex coordinates, expressed in floating point. Note that both EC and GM apply on demand refinement to permit the generation of a valid map. Therefore, this is not a verbatim copy of the original mesh from VOLMAP, but rather a (possibly) refined version of it;

- `model_input.txt`: contains the vertex coordinates of the mesh `model_input.mesh`, expressed as rational numbers. To avoid redundancy, the mesh topology is not replicated in this file;
- `model_output.mesh`: contains both the connectivity of the computed volumetric map and the mapped vertex coordinates, expressed in floating point. Mesh connectivity is the same as `model_input.mesh`;
- `model_output.txt`: contains the vertex coordinates of the mesh `model_output.mesh`, expressed as rational numbers. Once again, to avoid redundancy, the mesh topology is not replicated.

The models are packaged in `.zip` archives named `G*_EC.zip` and `G*_GM.zip`, containing the models of the various groups `G*` processed respectively with EC and GM, split into sub-folders based on the domain used for the mapping. The input and output `.mesh` files, containing models with floating point coordinates, can be loaded, visualized, and processed with the tools in VOLMAP [CL23]. Considering that working with rational coordinates and arithmetic is crucial in this field, as explained in the previous sections, we also provide three new functionalities to manage the models with rational coordinates (`.txt` files) introduced above.

- `load_RT`: loads a mesh representing an input model or a map with vertices in rational coordinates. Vertex coordinates are taken from the `.txt` file, while the connectivity from the corresponding `.mesh` file;
- `save_RT`: saves a mesh representing an input model or a map with vertices in rational coordinates into two files. Specifically, the rational vertex coordinates are saved into a `.txt` file, while the connectivity and the floating point coordinates are saved into a `.mesh` file;
- `map_checker`: takes as input a model and the corresponding volume map and checks the map's validity, in terms of number of flipped elements, considering both floating points and rational coordinates. This tool has a twofold usage: it allows to verify the correctness (bijectivity) of the map in exact rational coordinates, and it also allows to verify whether a floating point version of the same map retains bijectivity.

The aforementioned tools depend on CinoLib [Liv19] for basic mesh processing and detection of inverted elements in arbitrary precision. CinoLib, in turn, depends on the GMP [Gra91] library for this latter task. GMP does not need to be explicitly installed. CinoLib conveniently provides it by defining the compilation symbol `CINOLIB_USES_CGAL_GMP_MPFR`.

Time and hardware resources. The main goal of our experiments is to verify to what extent EC and GM can be used in realistic conditions where hardware and time resources are limited. We performed the experiments on a machine equipped with an Intel i9-7920X processor at 4.3 GHz with 12 physical cores (24 logic) and 128 GB of RAM. We always had 20 mappings running simultaneously with a time limit of 60 minutes each, and the RAM was shared between all the mappings..

4. Discussion

We report here on the results obtained with our experiments. As we stated before, we emphasize that this is by no meant to be intended as a comparison between Expansion Cones and Galaxy Maps, but rather as an attempt to understand to what extent these methods, which represent the state-of-the-art in the field, are currently capable of addressing practical needs that applications in computer graphics may desire. As it will become evident in the remainder of the section, due to the shortage of resources imposed in our setup, EC and GM converged on a valid map on largely disjoint sets of input models, making any attempt to directly compare these methods pointless.

Our analysis will be focused mainly on three aspects. Namely, the ability to complete the mapping task within the dedicated time budget, the amount of refinement introduced by the algorithms to ensure the existence of a map, and the ability to provide a result that not only is bijective when mesh coordinates are expressed exactly with rational numbers, but also when such numbers are rounded to the closest floating point, making the output result directly available to downstream applications.

Timeouts. As it could be easily predicted from the runtimes presented in the original papers, both EC and GM struggled to complete their mapping tasks within the given time budget, failing to succeed in a significant number of cases. Of the 486 input pairs tested, EC successfully produced a map in 11% of the cases, whereas the success rate for GM was 51%. The most challenging domain is undoubtedly the star-shaped for both methods, where failures occurred in almost all cases (Table 1). The lower success rate for EC could be explained by observing that this method always tries to compute the whole map from scratch. Conversely, thanks to its local approach, GM only needs to operate on the small portions of the mesh where the initial map (computed with Tutte [Tut63]) fails to be bijective, thus reducing the computational cost and also the likelihood to incur in slowdowns that are due to refinement or excessive complexity of the rational numbers used to express coordinates exactly. This local approach also makes the running time for GM hardly predictable, as it does not correlate with the global mesh size but rather with the amount and size of the individual clusters of inverted elements in the initial map (Figure 3). Nevertheless, the overall success rate remains insufficient for practical applications for both methods.

Mesh Growth. Both EC and GM ensure the existence of a valid map by refining the input mesh on demand. Excessive mesh refinement plays a central role in increased runtimes and memory pressure. Statistics and a visual plot of performances w.r.t. this aspect are showcased in Table 2 and Figure 4, respectively. The worst cases registered are an increase of the 132% for EC and 444% for GM. These numbers are comparable to the ones reported in the original articles.

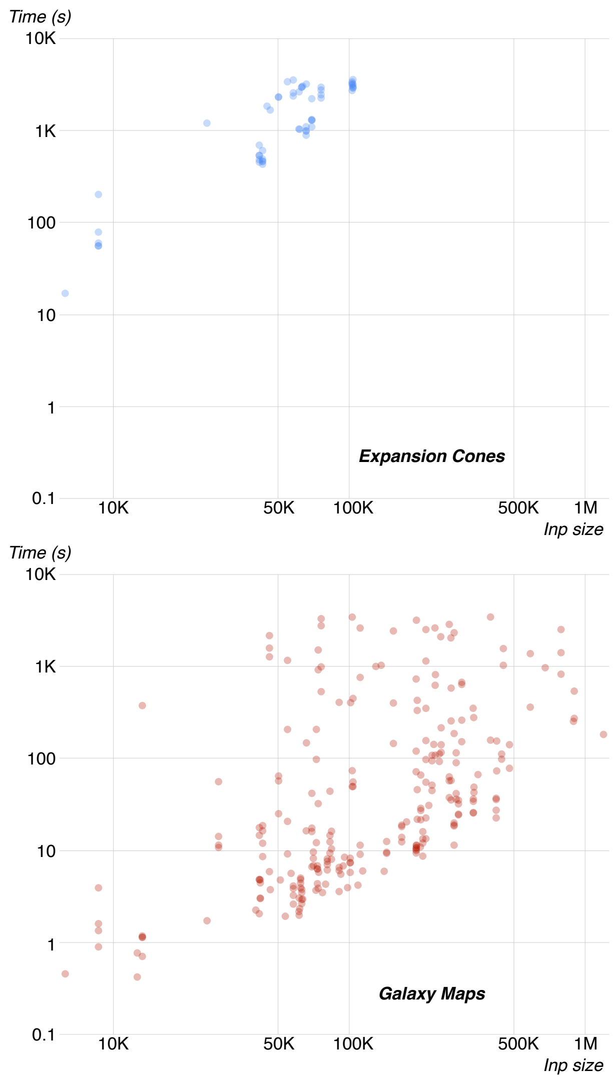

Figure 3: *In order to compute a valid map, EC operates globally on the input mesh, and, as you can see, its running time is proportional to the initial number of tetrahedra. Conversely, GM only needs to operate on a local cluster of flipped elements, making the running time unrelated to the input mesh size. In this plot, we show the mesh size on the x-axis and the running time on the y-axis.*

			Expansion Cones		Galaxy Maps	
Group	**Dom**	**#BC**	**#Maps**	**%**	**#Maps**	**%**
G1	cube	25	3	12%	24	96%
	tet	24	5	21%	20	83%
	pyr	25	1	4%	20	80%
	octa	24	2	8%	19	79%
	star	25	1	4%	1	4%
G2	cube	67	7	10%	49	73%
	tet	66	9	14%	24	36%
	pyr	66	8	12%	32	48%
	octa	67	7	10%	32	48%
	star	67	4	6%	0	0%
G5	sphere	26	7	27%	24	92%
G6	polycube	4	0	0%	4	100%
Total		*486*	*54*	*11%*	*249*	*51%*

Table 1: *For our experiments, we set a timeout of 60 minutes for each test. In this table, we report the number of tests for which the analyzed algorithms are able to produce a valid map (considering maps with rational coordinates) within the time limit. Specifically, the "#BC" column represents the number of boundary conditions in the VOLMAP dataset for each domain ("Dom") in every group G. The columns "#Maps" and "%" represent the number and the percentage of valid maps computed with respectively EC and GM.*

		Expansion Cones		Galaxy Maps	
Group	**Dom**	**#Maps**	**%growth**	**#Maps**	**%growth**
G1	cube	3	27/41/55%	24	1/10/72%
	tet	5	27/45/80%	20	2/50/313%
	pyr	1	132%	20	1/47/369%
	octa	2	21/-/68%	19	2/22/252%
	star	1	82%	1	444%
G2	cube	7	9/18/30%	49	1/16/98%
	tet	9	8/22/55%	24	8/17/134%
	pyr	8	7/20/54%	32	1/24/292%
	octa	7	8/20/47%	32	1/23/297%
	star	4	11/23/42%	0	-
G5	sphere	7	12/31/61%	24	0/3/10%
G6	polycube	0	-	4	2/3/4%

Table 2: *The two algorithms we tested both require performing refinement operations on the input models, which increment the original number of tetrahedra. In this table, we report the number of valid maps produced with EC and GM in the "#Maps" columns, while in columns "%growth" the corresponding percentage of growth of the tetrahedra in the input models. Specifically, we report the minimum, average, and maximum growth (min/avg/max%) for each domain ("Dom") in every group G.*

Floating Point Robustness. Recent literature in the field of surface mapping clearly shows that even the most robust algorithms may fail to provide a bijective map in floating point [SJZP19, FBRCA23, Liv24a, Liv24b]. Unconditional robustness can only be achieved if exact numerical models are used. The volume mapping methods we considered employ rational numbers to achieve this result [NCB23, HC23, CSZ16]. On the other hand, downstream applications are mostly incompatible with exact numerical models and require the map to be represented in floating point, requiring a lossy conversion that may potentially spoil the correctness of the result.

The overall number of cases in which a floating point version of the map was still bijective is reported in Table 3. As can be noticed, EC did not manage to produce a single valid floating map, whereas GM succeeded in a large number of cases (81.53%). As for the comments on the timeouts, we conjecture that such a big difference in the performances of the two algorithms is likely due to the fact that for GM most of the map comes from Tutte, and only a minority

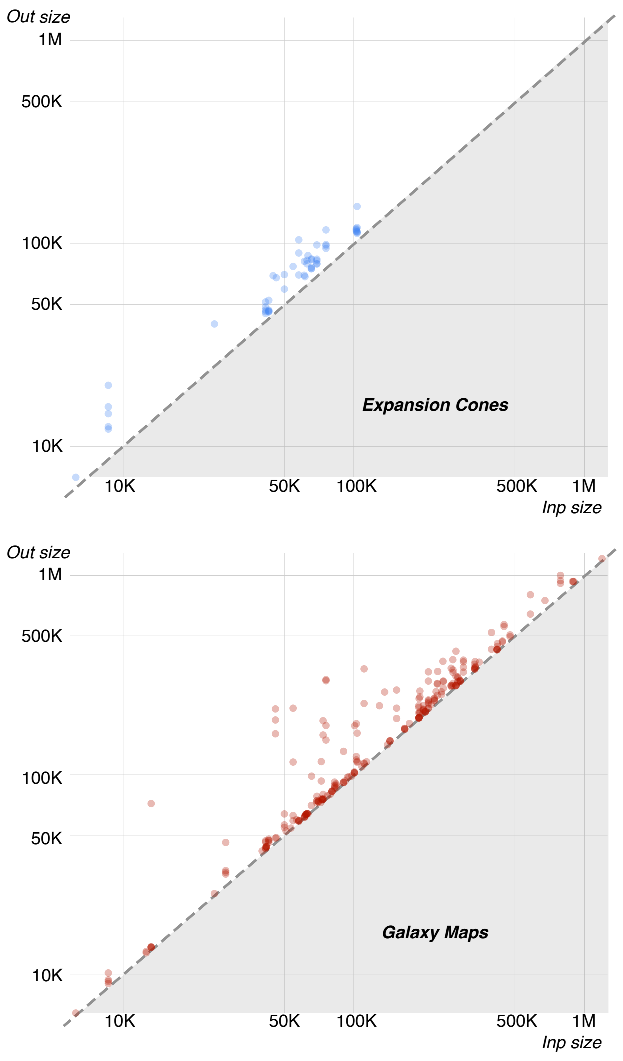

Figure 4: *The original size of the input mesh is increased by the refinement operations that both algorithms perform to create a valid map. This plot shows how the original mesh size (x-axis) can increase when computing the map (y-axis). Note that, to emphasize the analyzed behavior, we use a logarithmic scale in both axes, so even a tiny deviation from the main diagonal can represent a significant increment of the mesh size.*

of the mesh vertices are relocated to secure bijectivity. Conversely, EC needs to operate on all vertices, increasing the chances of triggering issues with the machine's precision.

It should be noted that, since both methods apply mesh refinement, not only the mapped mesh but also the input one may potentially contain inverted or vanishing elements (e.g., when an input edge is iteratively split multiple times). In our experiments, all maps but one contained inverted elements only in the target polyhedron. In one case (`buste4` model of the G2 group mapped on an octahedron domain), EC produced a refined version of the input mesh where more than 800 tetrahedra were flipped.

		Expansion Cones		Galaxy Maps	
Group	**Dom**	**#Maps**	**%FP valid**	**#Maps**	**%FP valid**
G1	cube	3	0%	24	96%
	tet	5	0%	20	95%
	pyr	1	0%	20	100%
	octa	2	0%	19	100%
	star	1	0%	1	100%
G2	cube	7	0%	49	90%
	tet	9	0%	24	100%
	pyr	8	0%	32	59%
	octa	7	0%	32	19%
	star	4	0%	0	-
G5	sphere	7	0%	24	100%
G6	polycube	0	-	4	100%

Table 3: *Both EC and GM need to perform refinement operations in the mesh structure. In order to avoid possible numerical errors due to the floating point numbers, they use rational arithmetic, increasing both computational time and memory requirements. In this table, we firstly report, for each domain ("Dom" column) in every group, the number of computed valid maps using rational arithmetic with the two algorithms ("#Maps"). Then, we report the percentage of corresponding maps ("%FP valid") that are still flip-free with the floating point representation.*

5. Conclusions and Future Works

We presented an extensive analysis of the two most prominent algorithms for the computation of provably bijective volume mappings: Expansion Cones [NCB23] and Galaxy Maps [HC23]. We tested their reference implementation on a sub-set of VOLMAP [CL23], a benchmark explicitly tailored to challenge this class of algorithms. Our analysis emphasizes that even though these methods provide formal guarantees of correctness, restricting them to operate on a limited time budget and memory dramatically decreases their success rate, to an extent that is hardly compatible with real applications.

Besides the concrete maps and statistics, we share with the community the maps we produced and the code necessary to parse and process such data, with the explicit intent to foster new research in the field to produce a new class of algorithms that match the formal guarantees of the existing ones but also complement them with efficiency and scalability. Results will be shared both in exact (rational) and floating point form, to also facilitate future comparisons in terms of compatibility with downstream applications that require a valid floating point map. Software facilities to operate with rational numbers are also included in the package we share.

The most natural extension of this work would be to expand the analysis to the remaining part of VOLMAP (i.e., groups G3 and G4), providing the community with up to 22k additional hypothetical maps. Besides this, it would be interesting to experiment alternative time limits (e.g., 2 hours, 5 hours), so as to understand the applicability of these algorithms in scenarios with different time constraints. Finally, we are aware of two forthcoming articles on volume mapping [HBC24, NCB24], which could be included in our future analysis and possibly improve the current results.

Acknowledgments

The authors are particularly grateful to Valentin Z. Nigolian and Steffen Hinderink for their valuable support with the code of the analyzed algorithms. F. Meloni acknowledges financial support through the scholarship "Tecnologie innovative e abilitanti per l'innovazione della pubblica amministrazione", founded by PNRR, DM 118/2022. G. Cherchi and R. Scateni gratefully 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, CUP: F23C24000240006. G. Cherchi also aknowledges the support to his activities by PRIN 2022 project "EUD4XR - End-User Development for eXtended Reality", funded by the Italian Ministry of University and Research (MUR) and by the European Union – NextGenerationEU, CUP: F53D23004380006.

References

[Ale23] ALEXA M.: Tutte embeddings of tetrahedral meshes. *Discrete & Computational Geometry* (2023), 1–11. doi:10.1007/s00454-023-00494-0. 2

[BC23] BRÜCKLER H., CAMPEN M.: Collapsing embedded cell complexes for safer hexahedral meshing. *ACM Transactions on Graphics 42*, 6 (2023), 1–24. doi:10.1145/3618384. 2

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

[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. 1, 2, 3, 4, 6

[CLSA20] CHERCHI G., LIVESU M., SCATENI R., ATTENE M.: Fast and robust mesh arrangements using floating-point arithmetic. *ACM Transactions on Graphics 39*, 6 (2020). doi:10.1145/3414685.3417818. 3

[CPAL22] CHERCHI G., PELLACINI F., ATTENE M., LIVESU M.: Interactive and robust mesh booleans. *ACM Transactions on Graphics 41*, 6 (2022). doi:10.1145/3550454.3555460. 3

[CSZ16] CAMPEN M., SILVA C. T., ZORIN D.: Bijective maps from simplicial foliations. *ACM Transactions on Graphics 35*, 4 (2016), 1–15. doi:10.1145/2897824.2925890. 2, 5

[DVPV03] DE VERDIERE É. C., POCCHIOLA M., VEGTER G.: Tutte's barycenter method applied to isotopies. *Computational Geometry 26*, 1 (2003), 81–97. doi:10.1016/S0925-7721(02)00174-8. 2

[FBRCA23] FINNENDAHL U., BOGIOKAS D., ROBLES CERVANTES P., ALEXA M.: Efficient embeddings in exact arithmetic. *ACM Transactions on Graphics 42*, 4 (2023). doi:10.1145/3592445. 2, 3, 5

[FL16] FU X.-M., LIU Y.: Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics 35*, 6 (2016), 1–12. doi:10.1145/2980179.2980231. 2

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer aided geometric design 14*, 3 (1997), 231–250. doi:10.1016/S0167-8396(96)00031-3. 2

[FPT06] FLOATER M. S., PHAM-TRONG V.: Convex combination maps over triangulations, tilings, and tetrahedral meshes. *Advances in Computational Mathematics 25* (2006), 347–356. doi:10.1007/s10444-004-7620-5. 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. doi:10.1007/s41095-021-0233-9. 2

[FW22] FARGION G., WEBER O.: Globally injective flattening via a reduced harmonic subspace. *ACM Transactions on Graphics 41*, 6 (2022), 1–17. doi:10.1145/3550454.3555449. 2

[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 40*, 4 (2021), 1–16. doi:10.1145/3450626.3459847. 3

[Gra91] GRANLUND T.: Gnu mp - the gnu multiple precision arithmetic library, 1991. URL: https://gmplib.org. 4

[HBC24] HINDERINK S., BRÜCKLER H., CAMPEN M.: Bijective volumetric mapping via star decomposition. *ACM Transactions on Graphics 43*, 6 (2024). doi:10.1145/3687992. 7

[HC23] HINDERINK S., CAMPEN M.: Galaxy maps: Localized foliations for bijective volumetric mapping. *ACM Transactions on Graphics 42*, 4 (2023). doi:10.1145/3592410. 1, 2, 3, 5, 6

[JSP17] JIANG Z., SCHAEFER S., PANOZZO D.: Simplicial complex augmentation framework for bijective maps. *ACM Transactions on Graphics 36*, 6 (2017). doi:10.1145/3130800.3130895. 2

[Lév24] LÉVY B.: Exact predicates, exact constructions and combinatorics for mesh csg. *arXiv preprint* (2024). doi:arXiv:2405.12949. 3

[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. doi:10.1007/978-3-662-59958-7_4. 4

[Liv20a] LIVESU M.: A Mesh Generation Perspective on Robust Mappings. In *Proceedings of 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* (2020). doi:arXiv:2012.08233. 2

[Liv24a] LIVESU M.: Advancing Front Surface Mapping. *Computer Graphics Forum* (2024). doi:10.1111/cgf.15026. 2, 3, 5

[Liv24b] LIVESU M.: Stripe embedding: Efficient maps with exact numeric computation. *ACM Transactions on Graphics 43*, 6 (2024). doi:10.1145/3687915. 2, 3, 5

[LYNF18] LIU L., YE C., NI R., FU X.-M.: Progressive parameterizations. *ACM Transactions on Graphics 37*, 4 (2018), 41. doi:10.1145/3197517.3201331. 2

[NCB23] NIGOLIAN V. Z., CAMPEN M., BOMMES D.: Expansion cones: A progressive volumetric mapping framework. *ACM Transactions on Graphics 42*, 4 (2023). doi:10.1145/3592421. 2, 3, 5, 6

[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 43*, 6 (2024). doi:10.1145/3687992. 7

[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. doi:10.1007/s10851-021-01038-y. 2

[PCS*23] PIETRONI N., CAMPEN M., SHEFFER A., CHERCHI G., BOMMES D., GAO X., SCATENI R., LEDOUX F., REMACLE J.-F., LIVESU M.: Hex-mesh generation and processing: a survey. *ACM Transactions on Graphics 42* (2023). doi:10.1145/3554920. 2

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Transactions on Graphics 36*, 4 (2017), 1. doi:doi.org/10.1145/2983621. 2

[SJZP19] SHEN H., JIANG Z., ZORIN D., PANOZZO D.: Progressive embedding. *ACM Transactions on Graphics 38*, 4 (2019). doi:10.1145/3306346.3323012. 2, 3, 5

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Transactions on Graphics 34*, 4 (2015), 1–9. doi:10.1145/2766947. 2

[SYLF20] SU J.-P., YE C., LIU L., FU X.-M.: Efficient bijective parameterizations. *ACM Transactions on Graphics 39*, 4 (2020), 111–1. doi:10.1145/3386569.3392435. 2

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

[WZC*23] WANG Q., ZHANG W.-X., CHENG Y.-Y., LIU L., FU X.-M.: Practical construction of globally injective parameterizations with positional constraints. *Computational Visual Media 9*, 2 (2023), 265–277. doi:10.1007/s41095-022-0269-5. 2

[YSLF20] YE C., SU J.-P., LIU L., FU X.-M.: Memory-efficient bijective parameterizations of very-large-scale models. In *Computer Graphics Forum* (2020), vol. 39, pp. 1–12. doi:10.1111/cgf.14122. 2

[ZGZJ16] ZHOU Q., GRINSPUN E., ZORIN D., JACOBSON A.: Mesh arrangements for solid geometry. *ACM Transactions on Graphics 35*, 4 (2016), 1–15. doi:10.1145/2897824.2925901. 3