STAG: Smart Tools and Applications in Graphics (2020)
S. Biasotti, R. Pintus and S. Berretti (Editors)

A Mesh Generation Perspective on Robust Mappings

Marco Livesu
CNR IMATI, Genoa

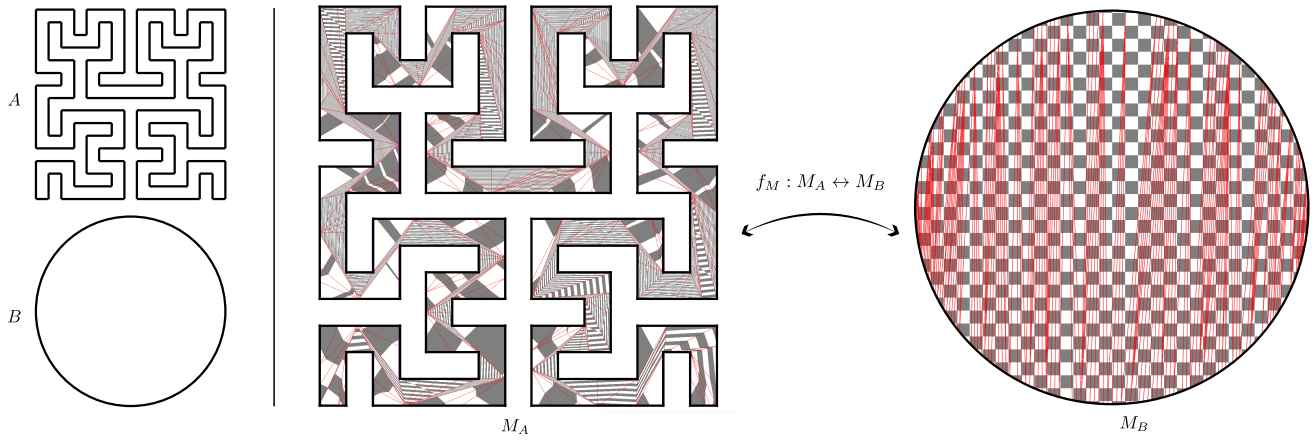

Figure 1: *We input the boundary representation of two topological spaces A and B, and output a compatible meshing of both domains. Meshes M_A and M_B have same connectivity and different embedding, thus defining a piece-wise linear bijective map $f_M : M_A \leftrightarrow M_B$.*

Abstract

Mapping a shape to some parametric domain is a fundamental tool in graphics and scientific computing. In practice, a map between two shapes is commonly represented by two meshes with same connectivity and different embedding. The standard approach is to input a mesh embedded in one domain plus a set of prescribed positions for its boundary vertices in the other domain, and compute the position of the interior points in the mesh. For the 2d case, there are numerous robust tools that follow this scheme. However, theoretical issues prevent them to scale to 3d domains, thus the robust generation of volumetric maps remains an important open scientific problem. Inspired by basic principles in mesh generation, in this paper we present the reader a novel point of view on mesh parameterization. We consider connectivity as an additional unknown, and assume that our inputs are just two boundaries that enclose the domains we want to connect. We compute the map by simultaneously growing the same mesh inside both shapes in an advancing front fashion. This change in perspective allows us to recast the parameterization problem as a mesh generation problem, granting access to a wide set of mature tools that are typically not used in this setting. Our practical outcome is a provably robust yet trivial to implement algorithm that maps non convex planar shapes to convex ones. Perhaps more interestingly, we speculate on possible extensions to planar maps between non convex domains, and to volumetric maps as well, listing the major challenges that arise. Differently from prior methods, our analysis leaves us reasonable hope that an extension to volumes is possible.

1. Introduction

A one-to-one map between two topological spaces A and B is a function $f : A \leftrightarrow B$ that connects points in both domains. When it comes to actual coding, the realization of this mathematical idea is typically implemented using simplicial meshes to represent topological spaces. Specifically, given two simplicial complexes M_A, M_B that *discretize* A and B, the piece-wise linear map f_M connecting them is implicitly defined by their shared connectivity. If both meshes do not contain degenerate elements, boundaries do not self intersect and all triangles have coherent orientation, these conditions are sufficient to grant a natural bijection $f_M : M_A \leftrightarrow M_B$ [Lip14]. In fact, any point $p \in M_A$ is identified by a mesh element and a unique set of barycentric coordinates that locate p inside it. Exploiting the shared connectivity, the image $\dot{p} = f_M(p)$ can be lo-

cated in M_B by considering the same mesh element and the same barycentric coordinates. Switching M_A with M_B allows to navigate the map in the opposite direction.

Algorithms for the parameterization of a given mesh M_A to some target domain B typically input M_A and the value of f_M for each boundary vertex $p \in \partial M_A$, such that they jointly interpolate the boundary ∂B. Then, these methods internally compute f_M for the interior vertices of M_A, completing the embedding of M_B. The most widely used strategy to accomplish this task amounts to define some energy – often encoding the distortion of the map – and then minimize it with numerical optimization. To this end, methods mainly differ for the types of energies they use and the numerical scheme used to minimize them.

In this paper we offer the reader a novel view on the mapping problem, which allows us to unlink it from this classical formulation based on numerical optimization. Keeping in mind that the ultimate goal is to produce two meshes M_A, M_B that discretize two target topological spaces and also share the same connectivity, we observe that an alternative way to formulate this problem consists in assuming as input two boundary representations of the spaces we want to connect, and to generate the wanted map by simultaneously *growing the same mesh* inside both domains. Differently from prior methods, in this case the amount of interior vertices and the way they are connected with each other and with the boundary vertices is not fixed a priori, but rather becomes an additional unknown. Although this may seem a complexification of the problem, it actually opens the space of solutions, allowing to design a mesh that is not tailored for one space only and then forcefully imposed in the other space, but is rather a good compromise for both spaces. Furthermore, this tiny shift in perspective allows us to recast the parameterization problem as a mesh generation problem, granting access to a wide set of mature tools that are typically not used in this setting.

In this article we set the basics principles of this idea, and demonstrate it in the context of planar maps. Specifically, we show that trivial polygon triangulation schemes can be used to initialize a bijective map between any simple polygon and a target convex domain. Solving the planar mapping problem may not seem particularly exciting, because solutions to this problem were known already in 1963, when Tutte published his famous article on barycentric mapping [Tut63]. The reason why we are still interested at studying novel solutions to this problem (and the main motivation for this work), is that known techniques do not extend to 3d. For the Tutte case this was shown multiple times via counter examples [CDL95, DVPV03], but even modern approaches such as Progressive Embeddings [SJZP19] raise major theoretical challenges going one dimension up (Section 2). As of today, the problem of initializing a valid simplicial map between two topologial spaces of dimension $d \geq 3$ is still open, and all we have is heuristic solutions that are based on numerical optimization of complex functionals, which do not provide guarantees of success [DAZ*20, SFL19].

We argue that our novel formulation of the mapping problem gives rise to a family of algorithms that *seem* to have potential to scale to higher dimensions. In Section 5 we dive a bit more into the details of possible extensions to 3d convex domains, and to maps between non convex domains as well, listing additional challenges that arise. Nevertheless, simplicial mesh generation is a rather mature field, with solid algorithms both for 2d and 3d. This gives us reasonable hope that such challenges could be overcome.

2. Background

In this section we discuss methods for robust surface mappings that are closest to us, also motivating why they cannot be extended to volumes.

The Tutte embedding [Tut63] was introduced in 1963 in the context of graph drawing, and was popularized in the graphics community by Floater in 1997, showing that any convex combination of neighbor vertices can be used to define the map [Flo97]. Ever since, a plethora of different methods have been proposed in the field, extending the original idea to topological tori and disk-like meshes with multiple boundaries [GGT06], as well as porting it to other spaces, such as Euclidean [AL15], hyperbolic [AL16] and spherical [AKL17] orbifolds. Despite numerous attempts, it was shown multiple times that the barycentric mapping does not extend to 3d. This is shown with a concise counter example in Figure 2; other failure cases are reported in [CDL95, DVPV03]. It is thought that under some assumptions that restrict the class of admissible graph topologies a 3d extension could still be possible [CDL95], but we are not aware of any success to this regard.

The Tutte embedding offers theoretical guarantees but – if pushed to the extreme – concrete implementations may fail to produce a valid mapping because of the limited precision of floating point systems. In [SJZP19] Shen and colleagues propose an alternative method, called Progressive Embeddings, which offers the same theoretical guarantees of Tutte, but is less sensitive to floating point implementations and also introduces less distortion in the map. The Progressive Embeddings algorithm is inspired by the progressive meshes concept [Hop96], and is based on the ability to deconstruct the topology of a triangle mesh by an ordered sequence of edge collapses, and to reconstruct the same mesh in another embedding with a sequence of vertex splits in the opposite order. Also this approach does not extend to 3d, the reason being twofold: (i) simplicial complexes in dimensions $d \geq 3$ may not be fully collapsible with a sequence of edge collapses, and even deciding whether a tetrahedral mesh is collapsible is NP Complete [Tan16, MF08, ADGL14]. Theory says that after a finite set of barycentric subdivisions any simplicial complex becomes collapsible [AB19], but still one should navigate the exponential space of all possible collapsing sequences to find a valid solution. Attempting to deconstruct a tetmesh along a heuristically computed collapsing sequence does not seem a good strategy either [LN19]. We personally tried many combinations of subdivisions and collapses, but always got stuck at some incollapsible configuration even on simple meshes (Figure 3); (ii) one may try to transform the input mesh into a mesh with different connectivity and known collapsing sequence via flip operators, but again this does not work because the graph of all possible triangulations of a given point set is connected for $d = 2$ [Law72, OB08], but there exist counter examples for the 3d case that show that it is disconnected for tetrahedralizations [DFM04]. All in all, these issues basically kill any hope that similar ideas could be extended to tetrahedral meshes.

A variety of methods offer the ability to perform a cross pa-

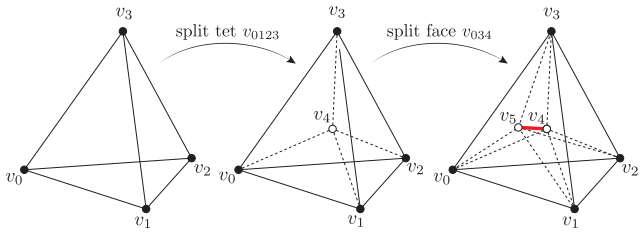

Figure 2: *Splitting a tetrahedron into four sub tets, and then splitting any of the so generated interior faces yields a simplicial complex with four boundary vertices (black) and two inner vertices (white). Keeping the boundary fixed, under the barycentric mapping proposed by Tutte [Tut63] the two inner vertices map to the same position. Therefore, the edge v_4v_5 (in red) and all the tets incident to it are collapsed, breaking the bijectivity of the map.*

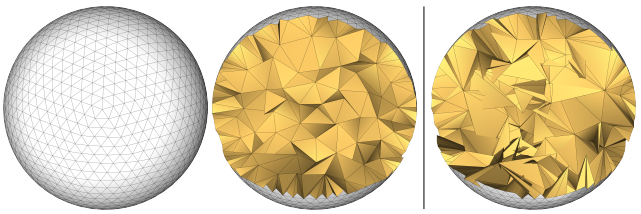

Figure 3: *Starting from a tetrahedralization of a sphere (left) and attempting to remove all interior edges along a random collapsing sequence yields a topology that cannot be further simplified (right), because any edge violates the link condition [DEGN98]. In the attempt to generate a collapsible mesh, any time we were stuck we also applied one step of barycentric subdivision, and then started again to collapse edges. After each iteration, the size of the incollapsible mesh we obtained was higher than the one at the iteration before.*

rameterization between two surface meshes of same topology. They employ composition of maps to intermediate domains such as coarse base complexes [KS04, SAPH04] or some polygonal schema [WZ14, Liv20, YZL*20]. These methods internally employ standard techniques (e.g. Tutte, or some derivation of it) to generate the underlying maps, therefore could potentially benefit from our contribution, and are orthogonal to it.

For the volumes, many methods formulate the mapping as an optimization problem. As for the 2d case, topology is fixed in input, and differences arise in the energies and numerical schemes used. We count mainly two family of approaches: (i) methods that input an invalid map and project it into the feasible space of solutions by fixing inverted elements [AL13, KABL14, SFL19, DAZ*20]; (ii) methods that input a valid solution – typically highly distorted – and iteratively improve the quality of the map by following the gradient of barrier energies that grow to infinity if an element becomes nearly degenerate or flips its orientation [RPPSH17]. The former do not provide any guarantee, because they may fail to even enter the feasible space. The latter guarantee the generation of a valid map if correctly initialized. In 2d the initialization step can be computed with [Tut63, SJZP19]. We are not aware of any 3d method that can provably generate a valid initial solution. Robustly initializing a volumetric map is the ultimate goal and main motivation for the mesh generation approach proposed in this paper.

A simplified version of the volume mapping problem was proposed in [CSZ16]. This method is based on foliations, and allows to map a genus zero simplicial mesh to a cube or a sphere. Users can select the type of foliation (e.g. radial for the mapping to a sphere) but cannot input a complete map between the surfaces using per vertex boundary conditions. Moreover, generating a valid piecewise linear map requires to perform aggressive mesh refinement, greatly increasing output mesh size even when mapping simple objects.

3. Method

In this section we introduce our novel surface mapping algorithm. We take as input two boundary representations of the domains A and B we want to connect, and we output a bijective function $f : A \leftrightarrow B$. Domain A must be a simply connected planar region, possibly containing concavities; domain B must be strictly convex. Both domains must come in the form of two closed chains of vertices, and are assumed to have the same number of elements, and not to contain any vanishing edge. Under these assumptions, there exists a one to one map $f_\partial : \partial A \leftrightarrow \partial B$ between them. Our goal is to extend such map from the boundary to the interior of both domains. The output of the algorithm amounts to two triangle meshes – M_A, M_B – that share the same connectivity, intrinsically defining a piece-wise linear map $f_M : M_A \leftrightarrow M_B$ that can be navigated as described in Section 1.

As briefly stated in the introduction, our key ingredient is a reformulation of the mapping problem in terms of mesh generation. In particular, we take inspiration from advancing front meshing algorithms such as [Löh96, MW95], which start from a boundary (or initial front) describing an empty region to be meshed, and obtain the output mesh by progressively attaching new elements to such front, until it completely vanishes. Our key observation is that if we consider as initial fronts two regions we want to connect, and we define a sequence of advancing moves that are compatible with both fronts, we will obtain the same meshing for the two domains, hence a mapping between them. Compared to classical approaches, our special setting imposes three important differences:

1) we work simultaneously in two domains, therefore advancing moves must be valid in both fronts, and must be applied following the same order;
2) at any time during execution, there must be a one to one correspondence between fronts in both domains, which must therefore be homotopic and contain the same number of vertices and edges;
3) assuming the absence of degenerate or flipped elements, *any* meshing is ok, regardless of the quality of its elements. This differentiates from classical mesh generation algorithms, which largely concern about per element quality

The first condition ensures that all mesh elements we introduce have their own linear map connecting their two copies in both domains. The second condition ensures that the algorithm does not

get stuck by creating topological mismatches in the fronts, which would prevent the completion of a valid global map. The third condition is just optional, but certainly applies to our case: we are only interested in initializing a valid map, without caring about geometric distortion. Methods that wish to generate a low distortion map may also leverage techniques for high quality advancing front mesh generation, but this is outside of the scope of this paper.

The algorithm works as follows: we initialize two fronts F_A, F_B with the two closed input chains of vertices defining the boundaries of A and B. Since each chain defines a simply connected polygon, we can use a simple triangulation scheme to advance the front, such as trivial earcut [Ebe08]. Specifically, we detect convex front vertices in F_A (i.e. vertices having inner angle lower than π), and check whether the triangle they form with their left and right neighbors contains any other vertex of F_A. If this is not the case, it means that the triangle is a valid ear, which can be *cut* (i.e. removed) from the front, yielding a simpler polygon with one vertex less. Whenever a valid ear is found, the corresponding vertex is removed from both F_A and F_B. Note that F_B is initialized with V_B, hence by our initial assumption is convex, and any of its vertices forms a valid ear. Moreover, any ear cut from it will preserve the convexity of F_B, because removing a point from a convex polygon yields a simpler convex polygon. This greatly simplifies the meshing process, because allows us to produce two meshes simultaneously by caring on the validity of each move only in one of the fronts (F_A). The iterative process continues as long as the size of the fronts is greater than 3. Once $|F_A| = |F_B| = 3$, we can complete the meshing by adding a triangle lid that vanishes both fronts. An algorithmic description of the method is given in 1. Figure 4 shows all the iterations for a simple example. Figure 1 shows a more complex mapping between a thickened space filling curve and a circle.

Convergence. Since everything is based on the triangulation of a simple polygon, and the ear test is performed only in F_A, the process is guaranteed to converge. This is ensured by the famous theorem that states that any polygon has at least two valid ears [Mei75], which is also the theoretical foundation for the earcut algorithm we employed.

Complexity. The complexity of the algorithm fully depends on the triangulation method of choice. For simplicity and ease of reproduction we opted for the earcut implementation present in Cinolib [Liv19]. Recent studies have shown that earcut is optimal (i.e. deterministic linear) on a restricted class of inputs [LCSA20], though on general simple polygons it has $\mathcal{O}(n^3)$ complexity if naively implemented, and can at most achieve $\mathcal{O}(n^2)$ with a smarter implementation [EET93, Ebe08]. The mapping algorithm we propose is not strictly linked to earcut, which could virtually be substituted with any other triangulation algorithm, obtaining a different asymptotic complexity.

Robustness. The whole method is based around well established algorithms, and is guaranteed to converge to a valid solution for any input that fulfills our input requirements. Since any planar polygon can be triangulated without additional points, our mapping does not necessitate to add vertices in the meshes along the way. This makes the whole procedure extremely robust also from an implementative point of view, because no approximations introduced by the floating point system are possible. Besides the input point coordinates – which are exact by definition – the only computations involving floating points regard the diagonal test to detect valid ears in F_A. To this end, exact point in triangle tests based on robust geometric predicates [She97] have been available in the meshing community since decades. All in all, this makes our algorithm robust without compromises, both theoretically and in practice.

ALGORITHM 1:

Input: two closed lists of vertices V_A, V_B, such that both chains form simple polygons, V_B is convex, and $|V_A| = |V_B|$.
Output: two triangle meshes $M_A(V_A, T)$ and $M_B(V_B, T)$ having same connectivity T, thus defining a bijection $f_M : M_A \leftrightarrow M_B$

$M_A = (V_A, \emptyset)$;
$M_B = (V_B, \emptyset)$;
initialize front F_A with V_A;
initialize front F_B with V_B;
while $|F_A| > 3$ **do**
  find an index i such that triangle v_{i-1}, v_i, v_{i+1} is a valid ear in F_A;
  insert triangle centered at i in both M_A and M_B;
  $F_A = F_A \setminus i$;
  $F_B = F_B \setminus i$;
end
fill the triangular hole in M_A with verts in F_A;
fill the triangular hole in M_B with verts in F_B;
return M_A, M_B;

4. Pushing the advancing front idea further

The declared ultimate objective of this research is to create a volumetric extension of the algorithm proposed in Section 3, in some way. In this section we speculate about two possible future directions, namely: a tentative extension of the earcut method to volumes, and the possibility to extend the mapping method to pairs of non convex domains, both in 2d and 3d.

4.1. Volumetric Earcut

In 2d the action of cutting an ear amounts to remove a convex vertex, substituting it with a straight segment that connects its two neighbors. Not only this operation is uniquely defined, but also decreases the inner angle of the neighbor vertices, preserving their convexity and, in turn, the convexity of the entire domain (if originally present). In 2d we deliberately exploit this property to restrict the quest for the next valid advancing move to one front only. For volumes, the situation is equivalent if and only if the convex vertex has valence 3. In such a case, cutting an ear amounts to delete the vertex, substituting it with a plane that interpolates its three neighbors, which is uniquely defined. This operation preserves the convexity of the domain as well, because dihedral angles can only decrease, and is the perfect dual of its 2d counterpart. Unfortunately, if the number of neighbors is higher than 3 (which is the most typical case for tetrahedral meshes) the operation of cutting an ear is not well defined, because the neighbor vertices may not be coplanar, and – even if they were – there would be multiple ways to triangulate them. One may still think to remove the vertex and triangulate the so generated pocket (some interesting ideas on how to tessellate these regions can be found in [CS94]), but if the neighbors are

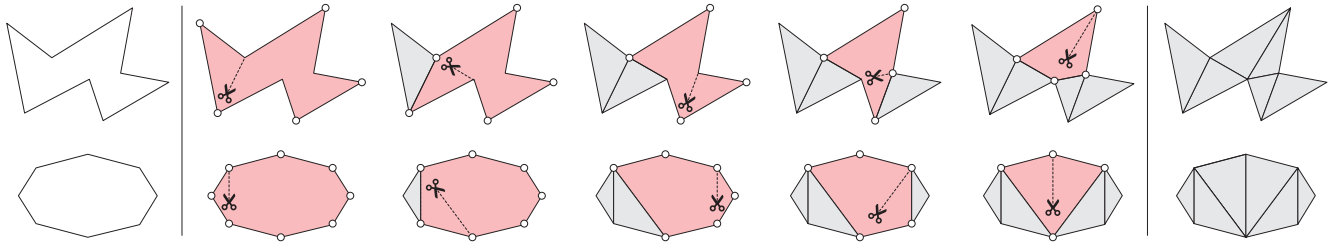

Figure 4: *Our pipeline starts with two simple planar polygons (left). We the initialize two fronts (boundaries of the red shaded areas), which we progressively tessellate by adding triangles centered at front vertices that form valid ears (white circles) in both domains. Upon convergence, we obtain two meshes with same connectivity and different embedding (right).*

not coplanar some of the possible tessellations will not preserve convexity, which is a crucial property for the algorithm. All in all, considering the non uniqueness of the earcut operation, the fact that some tessellations do not preserve convexity, and the fact that some other configurations may not be applicable in the starting (possibly non convex) domain, there may be configurations where the moves that are valid in one front are not valid for the other, causing a deadlock. We believe that the basic idea could still be ported to volumes, but a well defined ear cutting strategy that must be devised, possibly involving the insertion of new points in the domain, which ensure both convexity and uniqueness. To this end, it becomes crucial to devise methods that provably require a finite number of additional points – granting converge in a finite number of steps – and are also numerically stable (e.g. when defining the coordinates of the newly generated points).

4.2. Non convex domains

Another appealing extension of the mapping algorithm presented in Section 3 regards the ability to generate maps between two non convex shapes, either planar or volumetric. Interestingly, both extensions raise the same crucial challenge, which is the necessity to insert additional (Steiner) points in the domains to complete the meshing.

For the volumetric case vertex insertion is a classical problem, and indecomposable polyhedra that cannot be tetrahedralized without Steiner points were already known almost one century ago [Sch28]. Conversely, any planar domain can provably be triangulated without Steiner points [Mei75], but in our specific setting it is easy to verify that even two copies of a simple polygon with different vertex rotations do not admit a compatible remeshing, which instead can be found by inserting additional points inside the domain (Figure 5).

Considering its importance in the volumetric setting, vertex insertion has been widely studied and – from a theoretical standpoint – the topic is largely understood [GS15]. In practice, since computer programs often operate on numerical systems with finite precision, even the most advanced software available rely on heuristics for the computation of the coordinates of newly generated points, making this a critical step in any geometry processing pipeline [CLSA20].

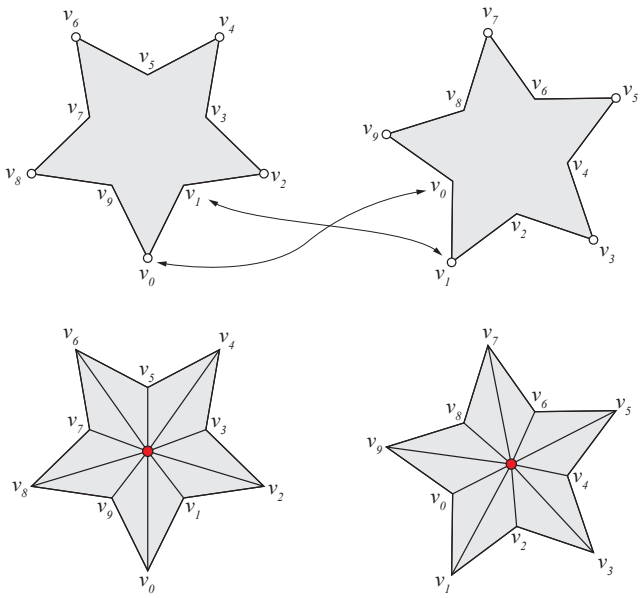

Figure 5: *Top: two copies of the same polygon with different vertex rotations may not admit a compatible remeshing. In this example each convex vertex in the left star maps to a concave vertex in the right star, and vice versa. Therefore, the intersection between valid ears (white circles) in the left and right polygons is empty. Bottom: inserting one point inside both domains (red circles) allows to triangulate both polygons with the same connectivity.*

The necessity to add points in the domain to complete the mesh heavily affects our approach, which in Section 3 assumed the convexity of one front across all iterations to greatly simplify the algorithm, permitting us to validate each move just in one of the two fronts. Advancing two fronts simultaneously in two non convex domains requires to always verify the validity of each move in both of them, with increased chances to get stuck in a deadlock configuration where no further move is possible. These cases can be unlocked by inserting additional Steiner points (Figure 5, bottom), but this re-opens a number of classical computational geometry questions that were answered for the meshing of a single domain and –

to the best of our knowledge – have no answer for the special case of compatible remeshing. Specifically:

- in case of a deadlock caused by two fronts that are indecomposable polyhedra, is it always possible to create a new move by adding one Steiner point in each domain? If not, what is the bound in the number of Steiner points that are necessary to grant the existence of a new valid move to advance both fronts simultaneously?

- what if the deadlock is caused by the empty intersection between the valid moves in the two fronts, but considering each front alone a move exists? Is this case analogous to the point before?

- what is the best positioning of a Steiner point in the context of simultaneous advancing front meshing? Can the coordinates of such point be expressed by rational numbers? This has huge practical importance, because it would guarantee that these points could be correctly positioned by a computer program (this is not the case for irrational numbers)

- is there a bound on the global number of Steiner points necessary to simultaneously triangulate two polyhedra? For single volumes, we know from theory that this number is bounded by $\mathcal{O}(1)$ from below (see the Schönhardt polyhedron [Sch28] and the subsequent generalization provided by Bagemihl [Bag48]), and by $\mathcal{O}(n^2)$ from above (by the Chazelle polyhedron [Cha84]). Should we expect similar bounds to exist also for pairs of shapes?

- can we guarantee that compatible advancing front meshing always convergences?

Looking at previous literature for the meshing of a single domain we are tempted to be optimistic about the existence of reasonable (i.e. polynomial) bounds in the number of Steiner points, and the possibility to always unlock a deadlock configuration in $\mathcal{O}(1)$. Nevertheless, precise and theoretically sound answers to all these questions should be provided in order to grant robust tools for the generation of volumetric or non convex planar mappings.

5. Conclusions

We have proposed a novel algorithm to robustly initialize a map between two simple polygons, when one of them is strictly convex. Our method is both theoretically sound and practically robust, but has little practical relevance because equally robust methods were already known for the 2d case [Tut63, SJZP19].

The interesting part (and main motivation) for this article is the proposal of a novel paradigm for the robust initialization of simplicial maps, which is rooted in the principles of advancing front mesh generation. We assume the input to be just a boundary representation of the domains to be connected, and we consider the mesh connectivity as an additional unknown. We then formulate the mapping as a mesh generation problem, where one wants to construct the same mesh in two different embeddings. This differentiates from classical approaches, which assume the topology of the mesh to be fixed, and solve the problem of positioning the interior vertices of a given mesh inside target domain.

We are mainly interested in this novel approach in the hope that similar ideas could be extended to volumes, because alternative robust 2d approaches such as barycentric mapping [Tut63] and Progressive Embeddings [SJZP19] cannot extend to 3d. To this end, we have shown that in case of planar maps between non convex shapes and also in the volumetric case additional challenges arise, and have provided a set of open questions that need to be answered by the computational geometry community to secure robust maps of this kind.

At the moment it is impossible to say whether a 3d extension would be computationally feasible, but the literature and wide availability of efficient tools for volume mesh generation leaves us reasonable hopes that in the near future a mapping method of this kind could be implemented.

References

[AB19] ADIPRASITO K., BENEDETTI B.: Barycentric subdivisions of convex complexes are collapsible. *Discrete & Computational Geometry* (2019), 1–19. 2

[ADGL14] ATTALI D., DEVILLERS O., GLISSE M., LAZARD S.: Recognizing shrinkable complexes is np-complete. In *European Symposium on Algorithms* (2014), Springer, pp. 74–86. 2

[AKL17] AIGERMAN N., KOVALSKY S. Z., LIPMAN Y.: Spherical orbifold tutte embeddings. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–13. 2

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

[AL15] AIGERMAN N., LIPMAN Y.: Orbifold tutte embeddings. *ACM Trans. Graph. 34*, 6 (2015), 190–1. 2

[AL16] AIGERMAN N., LIPMAN Y.: Hyperbolic orbifold tutte embeddings. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 1–14. 2

[Bag48] BAGEMIHL F.: On indecomposable polyhedra. *The American Mathematical Monthly 55*, 7 (1948), 411–413. 6

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

[Cha84] CHAZELLE B.: Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing 13*, 3 (1984), 488–507. 6

[CLSA20] CHERCHI G., LIVESU M., SCATENI R., ATTENE M.: Fast and robust mesh arrangements using floating-point arithmetic. *ACM Transactions on Graphics (SIGGRAPH Asia 2020) 39*, 6 (2020). doi:10.1145/3414685.3417818. 5

[CS94] CHAZELLE B., SHOURABOURA N.: Bounds on the size of tetrahedralizations. In *Proceedings of the tenth annual symposium on Computational geometry* (1994), pp. 231–239. 4

[CSZ16] CAMPEN M., SILVA C. T., ZORIN D.: Bijective maps from simplicial foliations. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–15. 3

[DAZ*20] DU X., AIGERMAN N., ZHOU Q., KOVALSKY S. Z., YAN Y., KAUFMAN D. M., JU T.: Lifting simplices to find injectivity. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 120–1. 2, 3

[DEGN98] DEY T. K., EDELSBRUNNER H., GUHA S., NEKHAYEV D. V.: Topology preserving edge contraction. In *Publ. Inst. Math.(Beograd)(NS* (1998), Citeseer. 3

[DFM04] DOUGHERTY R., FABER V., MURPHY M.: Unflippable tetrahedral complexes. *Discrete & Computational Geometry 32*, 3 (2004), 309–315. 2

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

[Ebe08] EBERLY D.: Triangulation by ear clipping. *Geometric Tools* (2008), 2002–2005. 4

[EET93] ELGINDY H., EVERETT H., TOUSSAINT G.: Slicing an ear using prune-and-search. *Pattern Recognition Letters 14*, 9 (1993), 719–722. 4

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

[GGT06] GORTLER S., GOTSMAN C., THURSTON D.: Discrete one-forms on meshes and applications to 3d mesh parameterization. *Computer Aided Geometric Design* (2006). 2

[GS15] GOERIGK N., SI H.: On indecomposable polyhedra and the number of steiner points. *Procedia Engineering 124* (2015), 343–355. 5

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 99–108. 2

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

[KS04] KRAEVOY V., SHEFFER A.: Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 861–869. 3

[Law72] LAWSON C. L.: Transforming triangulations. *Discrete mathematics 3*, 4 (1972), 365–372. 2

[LCSA20] LIVESU M., CHERCHI G., SCATENI R., ATTENE M.: Deterministic linear time constrained triangulation using simplified earcut. *arXiv preprint arXiv:2009.04294* (2020). 4

[Lip14] LIPMAN Y.: Bijective mappings of meshes with boundary and the degree in mesh processing. *SIAM Journal on Imaging Sciences 7*, 2 (2014), 1263–1283. 1

[Liv19] LIVESU M.: cinolib: a generic programming header only c++ library for processing polygonal and polyhedral meshes. *Transactions on Computational Science XXXIV* (2019). https://github.com/mlivesu/cinolib/. doi:10.1007/978-3-662-59958-7_4. 4

[Liv20] LIVESU M.: Scalable mesh refinement for canonical polygonal schemas of extremely high genus shapes. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2020). 3

[LN19] LOFANO D., NEWMAN A.: The worst way to collapse a simplex. *arXiv preprint arXiv:1905.07329* (2019). 2

[Löh96] LÖHNER R.: Extensions and improvements of the advancing front grid generation technique. *Communications in numerical methods in engineering 12*, 10 (1996), 683–702. 3

[Mei75] MEISTERS G. H.: Polygons have ears. *The American Mathematical Monthly 82*, 6 (1975), 648–651. 4, 5

[MF08] MALGOUYRES R., FRANCÉS A. R.: Determining whether a simplicial 3-complex collapses to a 1-complex is np-complete. In *International Conference on Discrete Geometry for Computer Imagery* (2008), Springer, pp. 177–188. 2

[MW95] MARCUM D. L., WEATHERILL N. P.: Unstructured grid generation using iterative point insertion and local reconnection. *AIAA journal 33*, 9 (1995), 1619–1625. 3

[OB08] OSHEROVICH E., BRUCKSTEIN A. M.: All triangulations are reachable via sequences of edge-flips: an elementary proof. *Computer Aided Geometric Design 25*, 3 (2008), 157–161. 2

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

[SAPH04] SCHREINER J., ASIRVATHAM A., PRAUN E., HOPPE H.: Inter-surface mapping. In *ACM SIGGRAPH 2004 Papers*. 2004, pp. 870–877. 3

[Sch28] SCHÖNHARDT E.: Über die zerlegung von dreieckspolyedern in tetraeder. *Mathematische Annalen 98*, 1 (1928), 309–312. 5, 6

[SFL19] SU J.-P., FU X.-M., LIU L.: Practical foldover-free volumetric mapping construction. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 287–297. 2, 3

[She97] SHEWCHUK J. R.: Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry 18*, 3 (1997), 305–363. 4

[SJZP19] SHEN H., JIANG Z., ZORIN D., PANOZZO D.: Progressive embedding. *ACM Trans. Graph. 38*, 4 (2019), 32–1. 2, 3, 6

[Tan16] TANCER M.: Recognition of collapsible complexes is np-complete. *Discrete & Computational Geometry 55*, 1 (2016), 21–38. 2

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society 3*, 1 (1963), 743–767. 2, 3, 6

[WZ14] WEBER O., ZORIN D.: Locally injective parametrization with arbitrary fixed boundaries. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–12. 3

[YZL*20] YANG Y., ZHANG W.-X., LIU Y., LIU L., FU X.-M.: Error-bounded compatible remeshing. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 113–1. 3