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

UV Parametrization via Topological Disk Segmentation of Surfaces

F. Maggioli[1] and S. Melzi[2]

[1]Pegaso University, Department of Information Sciences and Technologies, Italy
[2]University of Milano-Bicocca, Department of Informatics, Systems and Communication, Italy

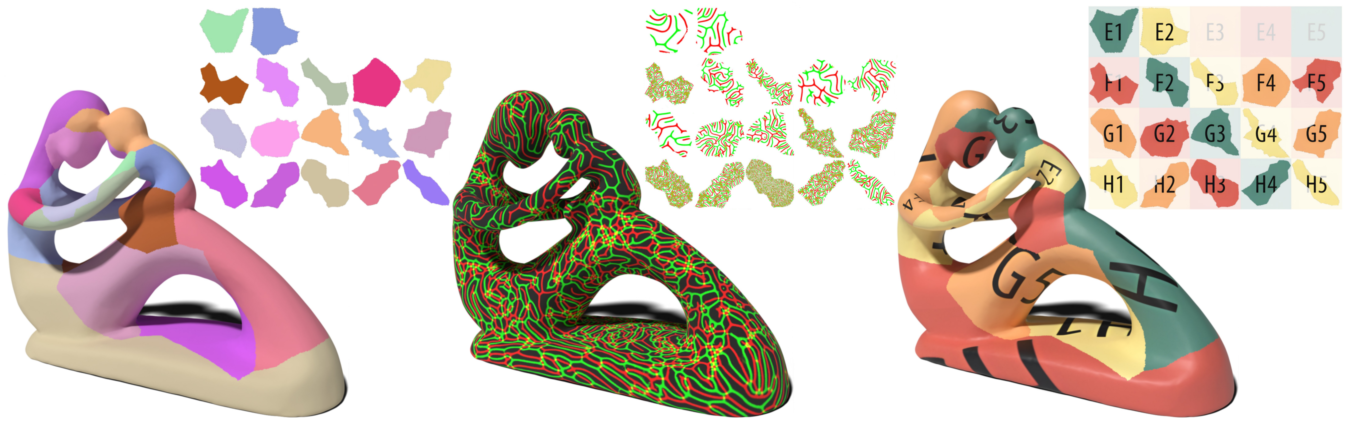

Figure 1: *Left: segmentation of the fertility model using our algorithm and the as-rigid-as-possible parametrization of the segments. Middle: the same global parametrization is used to compute a procedural texture for the surface via a slime simulation. Right: the parametrization is also used to apply a regular texture to the surface (the transparent portion of the texture is not mapped to the surface).*

Abstract

We present a reliable method for UV mapping that leverages a Voronoi-based decomposition of a triangulated surface mesh. Given a sparse set of sample points on the input shape, we construct the corresponding Voronoi partition and iteratively refine it to ensure that all regions are topologically equivalent to disks. The refinement proceeds in two stages: first, Voronoi cells are subdivided until disk-like topology is guaranteed; then, adjacent regions sharing substantial boundary portions are merged to reduce both their total number and perimeter-to-area ratio, while preserving disk equivalence. This topological guarantee enables straightforward and reliable UV parameterization. Our method exhibits an extremely low failure rate, making it suitable for practical use. In quantitative experiments on standard UV mapping benchmarks, we achieve performance comparable to state-of-the-art techniques. Furthermore, we analyze robustness and efficiency across different sampling densities, providing insights into the computational cost of each step of the pipeline.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Machine learning;*** *Shape analysis;* • ***Theory of computation*** $\rightarrow$ ***Computational geometry;***

1. Introduction

In computer graphics and geometry processing, triangular meshes have become the standard representation for three-dimensional surfaces. Their simplicity and expressiveness make them suitable for a wide range of applications, spanning from animation and physical simulation to shape analysis and texture mapping. Among these, UV mapping plays a central role, enabling the transfer of surface attributes such as textures and colors from a two-dimensional image domain to the surface of a 3D model. This is particularly beneficial for real-time and interactive applications, where performance requirements impose a low vertex count, and the high-quality standards force the use of attributes at sub-vertex resolution.

In the context of computer graphics [MYV93] and manufacturing design [PSOA18], significant efforts have been made to design recursive or repetitive patterns [NPP21, DGWB*14], curves [MNPP22, MMM*24], and shapes [RNP*22] directly on surfaces. However, most practical applications require more complex designs and higher efficiency, thus still relying on UV parametrizations that can be handcrafted by an artist, jointly processed with the surface for generating procedural patterns [Sta03, KCPS15, MMMR22], or baked with solid textures [EMP*02, Har01, MBMR22].

Despite its ubiquity and significance, the construction of reliable UV maps remains a challenging problem. Many existing methods are prone to failure in the presence of complex topologies, irregular sampling, or numerical instability [SSS22], making robustness and predictability critical requirements for practical use.

In this work, we introduce a novel approach to UV mapping that exploits the structural properties of triangular meshes to design a method that is efficient, simple to implement and highly reliable. The key idea is to build on a Voronoi decomposition of a sparse point sampling of the input mesh, and to refine this decomposition into regions that are guaranteed to be topologically equivalent to disks. Our algorithm proceeds in two phases. First, Voronoi cells are iteratively subdivided until all regions exhibit disk-like topology. Second, neighboring regions that share a substantial boundary component are merged to reduce both the total number of charts and their perimeter-to-area ratio, while maintaining disk equivalence. This topological guarantee allows for straightforward UV parameterization of each region and significantly reduces the occurrence of degenerate cases.

A primary motivation behind our work is to minimize the number of failure cases, which are a persistent obstacle in existing UV mapping pipelines. By leveraging mesh topology in combination with a principled refinement strategy, our method achieves an almost negligible failure rate while preserving performance comparable to state-of-the-art techniques. We further demonstrate that the algorithm is robust under varying sampling densities, and we provide a detailed analysis of its computational cost across the main stages of the pipeline.

The remainder of this paper is organized as follows. In Section 2, we review prior work on UV mapping and mesh decomposition. Section 3 presents the background necessary to introduce our approach. Section 4 describes the proposed algorithm in detail. Section 5 reports experimental results and comparisons with existing methods. In addition, in Section 6, we propose an analysis of the robustness of the proposed method to different densities of the mesh, as well as its runtime performance. Finally, Section 7 discusses limitations and outlines directions for future research.

2. Related work

The importance of constructing reliable UV maps dates back to the earliest days of computer graphics [Cat74] and makes it one of the most active research areas in geometry processing, even in recent years [BKP*10, Cra18]. Consequently, a wide variety of different approaches have been developed to accomplish the task of computing a UV parametrization of a surface [FSZ*21].

The standard pipeline to produce a UV parametrization consists of computing cuts of the surface and then parametrizing regions separately [CCC*08], although recent research efforts have shown that a joint optimization of the cuts and the parametrization can be an optimal strategy for computing reliable UV maps [PTH*17, LKK*18]. However, the task of cutting and parametrizing the surface at the same time can often result in challenging and unstable optimization problems, and thus the majority of the proposed solutions are focused on specific subproblems.

UV parametrization. Primarily, studies and efforts have focused on the computation of a piecewise linear map from a surface to the real plane, without concerns about segmenting or cutting the surface. The Tutte embedding [Tut63] was the first to be successfully applied to produce a fully bijective mapping (often also called *flip-free* mapping), in the case of surfaces with disk topology. Later, other approaches have been developed to achieve different goals. Well-established examples are: (i) the harmonic mapping, which minimizes distortion and metric dispersion [EDD*95], (ii) the conformal mapping, which locally preserves the angles [LPRM02], and (iii) the as-rigid-as-possible (ARAP) parametrization introduced by Liu *et al.* [LZX*08], which locally minimizes the length distortion. More sophisticated methods proposed refined energies and convoluted optimization problems that try to achieve fully bijective maps, while at the same time preserving some other desirable properties. A successful example is the Scalable Locally Injective Mapping (SLIM) [RPPSH17], which extends the ARAP metric with flip-free energies. Garanzha *et al.* [GKK*21] proposed a strategy to resolve flipped triangles, given an initial UV parametrization, by changing vertices positions. Another solution has been introduced by Gillespie *et al.* [GSC21], who propose to remesh the surface in order to compute a map that is both fully bijective and conformal. Although most of these approaches can be technically applied to arbitrary meshes, they are specifically designed to produce parametrizations of surfaces that are topologically equivalent to a disk, generally requiring a prior step where the mesh is either cut or segmented.

Surface segmentation. Our approach falls into the category of those methods that try to produce a valid segmentation of the surface, which then can be fed to a UV mapping algorithm for computing piecewise continuous parametrization of the entire surface. Earlier attempts revolved around the identification of feature points [KLT05] or regions with high curvature [SH02]. Other common strategies were based on the Voronoi decomposition from a sparse sample set [CCC*08], but also to deploy hierarchical structures [KT03], and the introduction of developability energies based on Gaussian curvature [JKS05]. Shapira *et al.* [SSCO08] proposed to introduce an energy based on the shape diameter function to obtain a consistent and informative segmentation of the surface. Spectral approaches based on the eigendecomposition of the Laplace-Beltrami operator were also very popular, primarily relying on K-means clustering [LZ04] and contour analysis [LZ07] in the spectral embedding. More recently, Mancinelli *et al.* [MM*23] developed in this direction by introducing a spectral pipeline to compute a semantically meaningful segmentation of a triangular mesh. Despite all these approaches successfully accomplishing their goal of segmenting a surface while highlighting or representing some types of features, they still present some limitations. On one side, most of the currently available solutions are based on the solution to numerical optimization problems, which makes them prone to failure and does not always guarantee the optimal solution. On the other hand, none of the discussed approaches is principled toward a decomposition into topological disks, which is a better fit for UV mapping algorithms.

3. Background

Continuous setting. In this work, we consider a surface as a 2-dimensional Riemannian manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$, possibly with boundary $\partial\mathcal{M}$. Given two points $x, y \in \mathcal{M}$ on the surface, we denote with $d_{\mathcal{M}}(x,y)$ their geodesic distance over $\mathcal{M}$. A segmentation of $\mathcal{M}$ is a set $S(\mathcal{M}) = \{S_1, \cdots, S_k\}$ of compact submanifolds of $\mathcal{M}$, where $\bigcup_{i=1}^{k} S_i = \mathcal{M}$ and $\forall i \neq j$, $S_i \cap S_j$ is either empty or a 1-dimensional manifold (*i.e.*, a curve). For deeper discussions about surfaces and Riemannian manifold, we remand to [DC16, Mor01].

The UV mapping (or parametrization) of a manifold $\mathcal{M}$ is a continuous bijective map $\varphi : \mathcal{M} \to \mathbb{R}^2$. When such a map cannot be found, we also admit as a valid UV parametrization a piecewise continuous bijective map $\varphi : \mathcal{M} \to \mathbb{R}^2$ defined by means of a segmentation $S(\mathcal{M})$ as

$$\varphi(x) = \varphi_i(x) \iff x \in S_i, \tag{1}$$

where $\varphi_i : S_i \to \mathbb{R}^2$ is a continuous bijective map (*i.e.*, a UV map for S_i). We remand to a recent survey from Fu *et al.* [FSZ*21] for additional details on UV parametrization.

Discrete setting. We discretize a 2-dimensional manifold $\mathcal{M}$ as a triangular mesh $M = (V, E, T)$, where: $V \subset \mathbb{R}^3$ is a set of vertices; $T \subset V \times V \times V$ is a set of triangles between vertices in V; $E \subset V \times V$ is a set of edges inferred from the triangles. Although we admit meshes with boundaries and self-intersections, throughout this work we require our meshes to be manifold, which means

- no more than two triangles can share a single edge;
- a vertex must be surrounded by a single triangle fan (either open or closed).

Given a triangular mesh $M = (V, E, T)$, as discussed in [Hir03], we can define the dual mesh of M denoted with $\hat{M} = (\hat{V}, \hat{E}, \hat{T})$, where:

- $\hat{v}_i \in \hat{V}$ is the *dual vertex* of the triangle t_i;
- $\hat{e}_{ij} = (\hat{v}_i, \hat{v}_j) \in \hat{E}$ is the *dual edge* of the edge shared by $t_i, t_j \in T$;
- $\hat{t}_i \in \hat{T}$ is the *dual face* of vertex $v_i \in V$.

For more details about dual meshes we refer to [Hir03].

Although we can define exact geodesic distances also on triangular meshes [MMP87], we discretize geodesic paths using shortest paths on the graph of the mesh. In particular, by exploiting Dijkstra's algorithm [Dij59], the computation of the distance $d_M(x, T)$ is numerically stable, and it is generally a fast and reasonably accurate approximation on discrete surfaces with a very high vertex count.

The UV parametrization of a triangular mesh $M = (V, E, T)$ is a piecewise linear map $J : M \to \mathbb{R}^2$ that maps each triangle $t_i \in T$ into a triangle in $\mathbb{R}^2$ by means of a linear map J_i. In the discrete setting, we can segment a triangular mesh by partitioning its triangles, and we can still produce a UV parametrization by mapping segments separately. We refer to the book by Botsch *et al.* [BKP*10] for a deeper discussion on geometric algorithms for triangular meshes.

4. Proposed method

Our approach for producing a UV parametrization of a mesh $M = (V, E, T)$ can be summarized in three major steps:

1. we produce a geodesic Voronoi decomposition of the surface;
2. we refine the decomposition to ensure regions are topological disks and have regular geometry;
3. we compute the UV mapping of each region and produce a UV parametrization for the entire surface.

4.1. Voronoi decomposition

As discussed in Section 3, a segmentation of a triangular mesh consists in a partitioning of its triangles. By considering the dual mesh $\hat{M} = (\hat{V}, \hat{E}, \hat{T})$, we lift the problem of partitioning triangles of the primal mesh to the partitioning of the vertices of the dual mesh. One common approach, which is also used by well-known geometry processing softwares like MeshLab [CCC*08], for partitioning surfaces is to compute the geodesic Voronoi decomposition. Given a set of sample vertices $P = \{p_1, \cdots, p_n\} \subset V$ on the surface, the geodesic Voronoi decomposition of $\hat{M}$ is a set $\text{Vor}(\hat{M}, P) = \{\text{Vor}(\hat{M}, p_1), \cdots, \text{Vor}(\hat{M}, p_n)\}$ such that

$$\text{Vor}(\hat{M}, p_i) = \left\{ v \ : \ p_i = \underset{p_j \in P}{\text{argmin}}\, d_{\hat{M}}(v, p_j) \right\}. \tag{2}$$

To ensure an even coverage of the surface and an efficient decomposition, we employ the algorithm presented by Maggioli *et al.* [MBRM25], that jointly computes a farthest point sampling of n sample points and their Voronoi decomposition in time $\mathcal{O}\left(|\hat{V}| \log |\hat{V}| \log n\right)$.

4.2. Decomposition refinement

Subsampling. A geodesic Voronoi decomposition provides a valid segmentation of the surface, but it does not provide any guarantee about the topology of Voronoi regions (see the dark blue region on the left frame in Figure 2), as instead required by our UV mapping solution. Liu *et al.* [LFXH17] have shown that, with sufficient sampling density, Voronoi regions can be guaranteed to be topologically equivalent to a disk; however, their approach results to be computationally inefficient and produces a number of regions that quickly becomes incontrollable. In contrast, we consider the fact that a compact surface $\mathcal{M}$ (continuous or discrete) is a topological disk if and only if its Euler characteristic $\chi(\mathcal{M}) = 1$.

This notion can be exploited to refine the initial sampling and produce a Voronoi decomposition into topological disks. As mentioned earlier, with enough sample points, Voronoi regions are guaranteed to be topologically equivalent to a disk. However, instead of increasing the sampling density everywhere, we consider the fact that some Voronoi regions may already satisfy this condition. Consequently, we only need to further decompose the other regions. Consider the sample p_i and the Voronoi region $\text{Vor}(\hat{M}, p_i)$

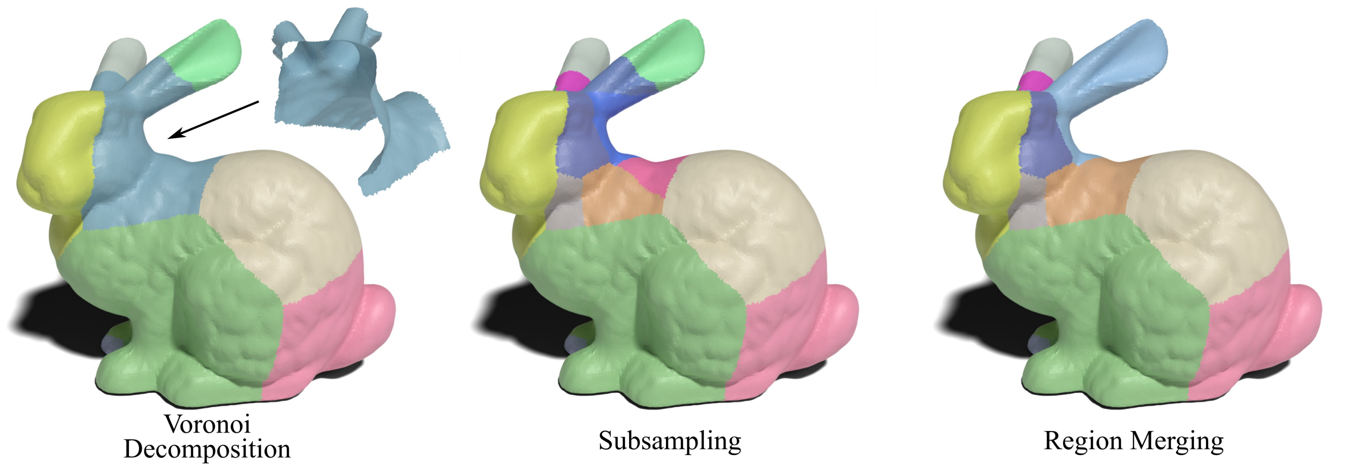

Figure 2: *Left: the geodesic Voronoi decomposition of the surface produces regions that are not topologically equivalent to a disk. Middle: our first refinement step adds sample points to the critical areas, ensuring that each Voronoi region is a topological disk. Right: our greedy merging algorithm minimizes the number of regions in the decomposition, while still ensuring a segmentation made of topological disks.*

Algorithm 1 Subsampling algorithm.

```
 1: procedure SUBSAMPLING($\hat{M}$, $P$, $k$)
 2:     $S \leftarrow \emptyset$
 3:     for $p_i \in P$ do
 4:         $M_i \leftarrow$ submesh induced by $\mathrm{Vor}(\hat{M}, p_i)$
 5:         if $\chi(M_i) = 1$ then
 6:             $S \leftarrow S \cup \{M_i\}$
 7:         else
 8:             $P' \leftarrow$ VORONOIFPS($M_i$, $k$)
 9:             $S_i \leftarrow$ SUBSAMPLING($M_i$, $P'$, $k$)
10:             $S \leftarrow S \cup S_i$
11:         end if
12:     end for
13:     return $S$
14: end procedure
```

associated with the sample. If such region is topologically equivalent to a disk, we leave it as is. Otherwise, we consider the submesh $M_i = (V_i, E_i, T_i) \subset \hat{M}$ induced by $\mathrm{Vor}(\hat{M}, p_i)$, and we recursively call the SUBSAMPLING procedure to apply a Voronoi decomposition. The procedure is summarized in Algorithm 1, and produces a valid decomposition into topological disks (see the middle frame of Figure 2).

Proposition 1 Given a mesh $\hat{M} = (\hat{V}, \hat{E}, \hat{T})$ and a set of sample points P, the procedure described in Algorithm 1 terminates and produces a decomposition of $\hat{M}$ into topological disks.

Proof If $\mathrm{Vor}(\hat{M}, p_i)$ is a topological disk for all p_i, then the condition at Line 5 is always met and the recursion is never called. Consequently, the algorithm terminates and S contains a decomposition of $\hat{M}$ into topological disks.
Otherwise, let p_i such that $\mathrm{Vor}(\hat{M}, p_i)$ is not a topological disk, and let us consider the induced submesh $M_i = (V_i, E_i, T_i)$. If $|V_i| \leq k$, the mesh is decomposed into its single vertices, which induce topological disks by definition of manifold mesh: M_i is decomposed into topological disks and the algorithm continues to the following iteration. If $|V_i| > k$, the procedure is called recursively on M_i. However, since $M_i \subset \hat{M}$, then $|V_i| < |V|$, meaning that each level of recursion reduces the size of the mesh. Eventually, the algorithm will produce a mesh $M^* = (V^*, E^*, T^*)$ such that $|V^*| \leq k$ and the recursion will terminate.
Each recursive step produces a decomposition of the submesh into topological disks. Since the union of all these regions will result in the original mesh $\hat{M}$, the algorithm successfully produces a decomposition of $\hat{M}$ into topological disks. □

Region merging. As shown in the middle plot of Figure 2, the subsampling procedure could produce a large amount of small regions. While this makes it easier for UV mapping algorithms to locally parametrize the surface, the dramatic fragmentation of the resulting global parametrization could virtually have no practical use.

To overcome this issue, we introduce a greedy strategy for merging adjacent regions iteratively, while at the same time maintaining a decomposition into topological disks. Our approach takes advantage of the inclusion–exclusion principle for the Euler characteristic with compact support in locally compact space, that is

$$\chi(\mathcal{M} \cup \mathcal{N}) + \chi(\mathcal{M} \cap \mathcal{N}) = \chi(\mathcal{M}) + \chi(\mathcal{N}) \,. \tag{3}$$

For each pair of adjacent regions M_i and M_j, we can easily verify if their boundary $M_i \cap M_j$ is a topological segment (*i.e.*, $\chi(M_i \cap M_j) = 1$). By recalling that $\chi(M_i) = \chi(M_j) = 1$, it follows that the union region $M_i \cup M_j$ is a topological disk (*i.e.*, $\chi(M_i \cup M_j) = 1$) iff $\chi(M_i \cap M_j) = 1$. By taking advantage of appropriate data structures like hashmaps and hashsets, these conditions are efficiently verifiable.

The regions in our decomposition should also have a geometry that simplifies the UV parametrization as much as possible. This means that regions with an elongated shape (*e.g.*, long triangle strips) or presenting choke points (*e.g.*, two large regions connected by a single triangle) should be avoided as much as possible. To mitigate similar situations, we introduce a score system for merging candidates, based on the relation between their area and their boundary. In particular, let A_i and A_j be, respectively, the areas of M_i and M_j, and let ℓ_{ij} be the length of their boundary $M_i \cap M_j$. For 2-dimensional shapes, the area is related to the square of the perimeter, and thus we define our score for the merging of M_i and M_j as

$$\vartheta_{ij} = \frac{\ell_{ij}}{\sqrt{\max\left(A_i, A_j\right)}}. \tag{4}$$

We propose a greedy merging algorithm, which is summarized in Algorithm 2 and proceeds as follows. Firstly, we compute the dual graph of the segmentation, where each node represents a region and each edge represents the adjacency between regions. We the filter out all the edges between regions whose union will not result in a topological disk, and for all the other edges we compute the score as described in Equation (4). We also use a threshold ε for the score to exclude all pairs of regions that would merge along a short boundary (and thus form undesirable geometries like choke points or elongated shapes). If there are edges left, we select the edge with the highest score and merge the nodes in the graph and the corresponding regions. By preferring the merging of regions that share a longer boundary with respect to their area, we guide the algorithm toward a minimization of the total perimeter-to-area ratio, resulting in the formation of more regular shapes. This whole process is iterated until the graph is left unchanged, meaning no more regions can be merged.

The application of Algorithm 2 produces a smaller number of larger regions, reducing the fragmentation of the decomposition and resulting in a more accessible segmentation. The example in the right frame in Figure 2 shows how small regions such as those on the ears can be merged to simplify the decomposition while still preserving the disk topology.

4.3. Parametrization

The major advantage of our approach, is that it is not bounded to a specific UV parametrization algorithm. Indeed, our method produces a segmentation of the mesh into topological disks, which are easier to parametrize using well-know methods like harmonic mapping [EDD*95], conformal mapping [LPRM02], and as-rigid-as-possible mapping [LZX*08]. Consequently, our method can be plugged in any parametrization pipeline relying on a segmentation of the surface [Tut63, RPPSH17, GKK*21], but it can also be injected into algorithms that jointly perform segmentation and parametrization [GSC21].

To produce the final parametrization, in our implementation we use a naive packing algorithm that arranges the k regions in a square grid of side $p = \left\lceil \sqrt{k} \right\rceil$. We also search for the optimal rotation that maximizes the total area of each region when it is rescaled to fit the bounds of its cell. Although this approach does not produce an optimal packing, it is very efficient and can be easily replaced with more optimized algorithms.

Algorithm 2 Region merging algorithm.

```
 1: procedure RegionMerging($M$, $S$, $\varepsilon$)
 2:     $\mathcal{G} = (\mathcal{V}, \mathcal{E}) \leftarrow$ dual graph of segmentation $S$
 3:     while $\mathcal{G}$ is unchanged do
 4:         $Q \leftarrow$ empty priority queue
 5:         for $(S_i, S_j) \in \mathcal{E}$ do
 6:             if $\chi(S_i \cap S_j) = 1$ then
 7:                 $\vartheta_{ij} \leftarrow$ Score($M$, $S_i$, $S_j$)
 8:                 if $\vartheta_{ij} \geq \varepsilon$ then
 9:                     $Q$.push($\vartheta_{ij}, (S_i, S_j)$)
10:                 end if
11:             end if
12:         end for
13:         if $|Q| > 0$ then
14:             $(S_i, S_j) \leftarrow Q$.pop()
15:             Merge $S_i$ with $S_j$ and update $\mathcal{G}$
16:         end if
17:     end while
18: end procedure
```

4.4. Parameter tuning

Our entire pipeline requires the definition of three parameters, namely:

n: the number of samples for the Voronoi decomposition;
k: the number of samples for the subsampling algorithm;
ε: the score threshold for merging adjacent regions.

In our implementation, we use $n = 10$ as default, since this is the same value used in MeshLab for the UV parametrization via Voronoi decomposition [CCC*08]. We also set $k = 5$, as in most of our tests, this value was sufficient to decompose the regions into topological disks with a single recursion step, while at the same time it did not dramatically increase the number of regions. Finally, instead of performing an exhaustive search to set the threshold ε, we consider that for regular polygons with N sides, Equation (4) evaluates to

$$\varepsilon_N = \frac{2}{\sqrt{N \cot\left(\frac{\pi}{N}\right)}}. \tag{5}$$

We test our method with $\varepsilon = \varepsilon_N$ for different values of N on the parametrization benchmark proposed by Shay *et al.* [SSS22], and we record that the best overall results are achieved in the setting $\varepsilon = \varepsilon_5 \approx 0.76$. Figure 3 depicts an example decomposition with different threshold values.

5. Results

We implemented our method in C++, using Eigen [GJ*10] and libigl [JPS*13]. The source code is available at `https://github.com/filthynobleman/disk-segmentation`.

For the UV parametrization step, we consider the harmonic mapping algorithm proposed by Eck *et al.* [EDD*95], eventually re-

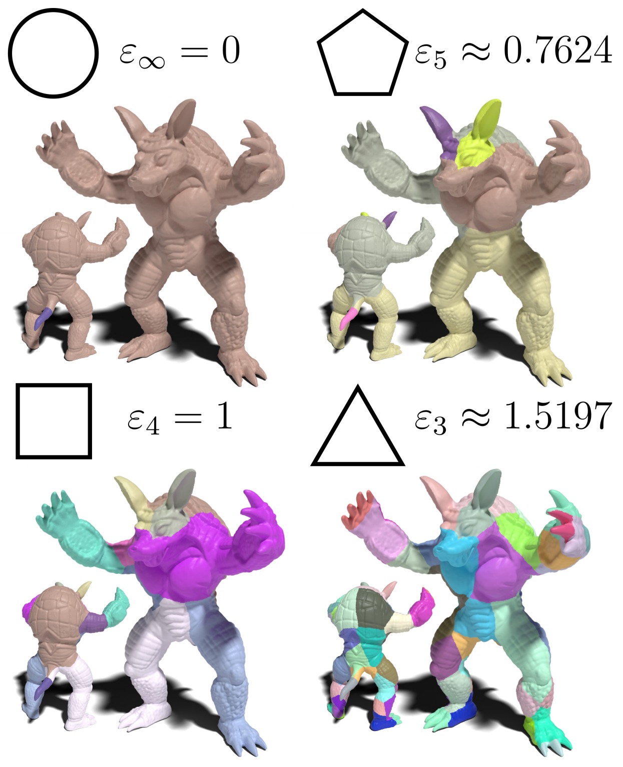
$\varepsilon_\infty = 0$
$\varepsilon_5 \approx 0.7624$
$\varepsilon_4 = 1$
$\varepsilon_3 \approx 1.5197$

Figure 3: *Different segmentations of a mesh produced by our method at different values of the score threshold ε corresponding to different polygons.*

fined by the as-rigid-as-possible parametrization introduced by Liu *et al.* [LZX*08], as well as the conformal mapping technique presented by Lèvy *et al.* [LPRM02]. Using these three parametrization techniques, we compare our segmentation approach with the naive geodesic Voronoi segmentation (Voronoi) [CCC*08], the shape diameter segmentation proposed by Shapira *et al.* (Shape Diameter) [SSCO08], and the segmentation algorithm based on spectral properties introduced by Mancinelli *et al.* (Spectral) [MM*23]. Additionally, we also consider a segmentation into connected components as a naive baseline (Conn. Comp.). For reference, we also report the results of the OptCuts algorithm introduced by Li *et al.* [LKK*18], which jointly optimize for a segmentation and a UV parametrization of the surface.

We conduct our experiments on the parametrization benchmark proposed by Shay *et al.* [SSS22]. With this work, the authors introduced a dataset for mesh parametrization that comprises ~6.5k surfaces of different topologies and geometries (the *Uncut* category), plus ~12k surfaces with cuts already provided (the *Cut* category). Each shape comes with a UV parametrization provided by an artist, and the authors did also provide a benchmarking application for evaluating algorithms; we use such application for our quantitative analysis. Since we focus on segmentation algorithms, we select the manifold meshes of the *Uncut* category. We summarize the results of our comparisons in Table 1, where for each segmentation and

	segmentation	area dist.	angle dist.	flipped triangles	cut length	failure rate
Harmonic	Conn. Comp.	0.71	1.44	2.98%	0	45.71%
	Shape Diam.	0.85	1.44	3.45%	0.43	50.99%
	Spectral	0.99	1.98	2.07%	11.54	58.28%
	Voronoi	0.75	1.39	1.03%	13.53	3.15%
	Ours	0.87	1.08	1.47%	3.11	0.04%
Conformal	Conn. Comp.	0.51	0.79	8.88%	0	63.11%
	Shape Diam.	0.63	0.96	10.41%	0.33	68.29%
	Spectral	0.79	0.77	7.18%	6.11	85.61%
	Voronoi	0.42	0.27	2.56%	7.58	87.26%
	Ours	0.78	0.56	4.90%	1.78	44.38%
ARAP	Conn. Comp.	0.32	0.90	8.27%	0	45.04%
	Shape Diam.	0.43	1.01	9.26%	0.40	51.27%
	Spectral	0.49	0.51	3.04%	11.54	58.28%
	Voronoi	0.45	0.36	2.62%	13.53	3.16%
	Ours	0.59	0.77	5.62%	3.11	0.09%
	OptCuts	0.05	0.13	~0%	1.47	12.09%

Table 1: *Results on the parametrization benchmark [SSS22] on the manifold shapes in the* Uncut *category. Segmentation methods are compared under the same parametrization algorithm.*

	segmentation	area dist.	angle dist.	flipped triangles	cut length	failure rate
Harmonic	Conn. Comp.	1.16	2.14	7.43%	0	74.48%
	Shape Diam.	1.25	1.92	6.52%	0.83	68.63%
	Spectral	0.99	1.98	2.07%	11.54	58.28%
	Voronoi	0.83	1.48	1.47%	16.15	0.32%
	Ours	0.99	1.22	1.94%	4.63	0%
Conformal	Conn. Comp.	0.84	1.57	19.88%	0	83.20%
	Shape Diam.	0.98	1.67	2.23%	0.81	82.25%
	Spectral	0.79	0.77	7.18%	6.11	85.61%
	Voronoi	0.45	0.35	3.42%	7.77	84.69%
	Ours	0.96	0.71	6.34%	2.90	50.01%
ARAP	Conn. Comp.	0.50	1.42	15.65%	0	73.30%
	Shape Diam.	0.65	1.46	15.12%	0.75	68.84%
	Spectral	0.49	0.51	3.04%	11.54	58.28%
	Voronoi	0.47	0.44	3.72%	16.15	0.34%
	Ours	0.68	0.90	7.06%	4.63	0.02%
	OptCuts	0.05	0.15	~0%	2.12	17.27%

Table 2: *Results on the parametrization benchmark [SSS22] on the manifold shapes in the* Uncut *category, only considering shapes that have non-disk topology. Segmentation methods are compared under the same parametrization algorithm.*

parametrization method, we report the area and angle distortion, the percentage of flipped triangles, and the length of the cut performed by the segmentation algorithm. All the results are averaged on the cases for which the methods successfully find a parametrization. The percentage of failure cases is also shown in the last column. We additionally report the results of a similar comparison in Table 2, where we only consider the more challenging subset of meshes that do not have disk topology.

We see from the tables that the connected component segmentation, the shape diameter approach, and the method based on spectral properties produce segmentations that cannot be parametrized

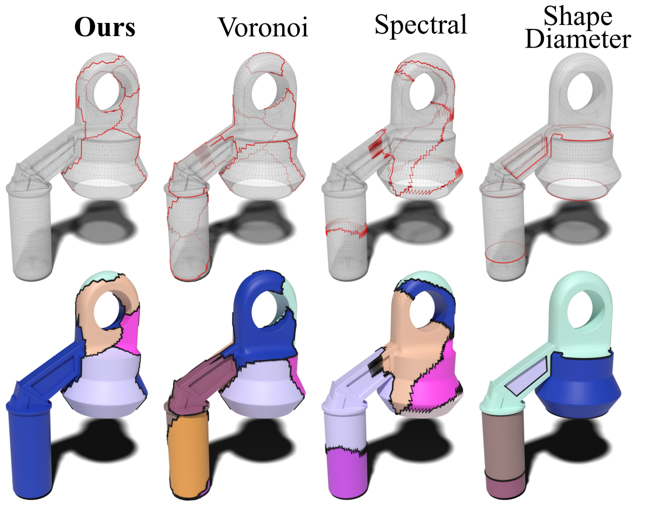

Figure 4: *Comparison between the cuts along the mesh introduced by the different segmentation algorithms on an object in the parametrization benchmark dataset from Shay et al. [SSS22]. Top row: objects are rendered in slight transparency to show the cuts on the other side. Bottom row: regions are colored differently to highlight the segmentation.*

in a significant number of cases. Interestingly, when using the conformal mapping algorithm, the failure rate for all the methods increases drastically because the eigensolver fails to converge. By comparing the results for conformal mapping between Tables 1 and 2, we can see that our approach fails in a significantly lower percentage of cases and that meshes without disk topology are more challenging; this suggests that disk decomposition simplifies the parametrization by favoring the convergence of the eigensolver.

Another property in which our approach achieves good performance is the total length of the cuts along the mesh to produce a segmentation. Due to the region merging strategy introduced in Section 4, our method is able to drastically reduce the amount and the total length of cuts needed by the simple Voronoi decomposition highlighting the actual contribution of our pipeline. Similarly, when the method based on spectral properties produces a high number of critical points, the number of regions is not optimized. Furthermore, this latter strategy produces a vertex-based partitioning; when we convert it to a triangle-based partitioning, some jagged edge lines are produced that further increase the total length of the cut (see Figure 4). From Figure 4, we can also appreciate that the shape diameter approach tends to produce clean cuts that are aligned with sharp features of the object, resulting in shorter cut lengths, on average (see Tables 1 and 2). Nevertheless, this approach is unaware of the topological information of the regions it produces; this results in a large number of non-disk regions that are more challenging to parametrize, and consequently in a significantly larger failure rate in the parametrization of the surface.

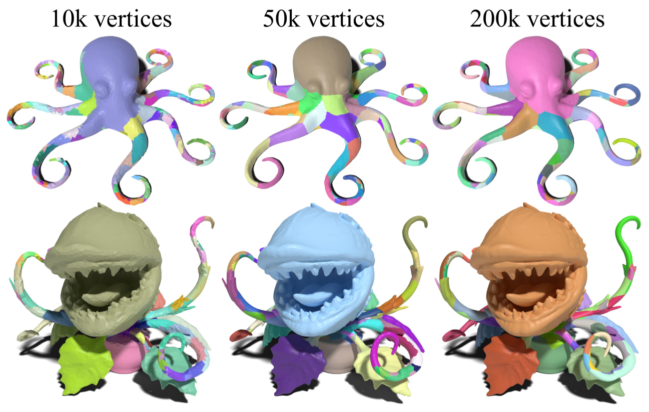

Figure 5: *Our disk decomposition algorithm applied to meshes at different resolutions, using the same set of starting Voronoi samples. For the octopus, we set the merging threshold to $\varepsilon = \varepsilon_4$, while for the chomper we set it to $\varepsilon = \varepsilon_5$.*

6. Analysis

A major advantage of our approach for shape segmentation is that it is fully combinatorial. By not involving numerical steps in our procedure, we are always able to produce a valid decomposition of the surface, independently of the geometry and connectivity of the mesh. This, combined with the fact that the regions are all topologically equivalent to a disk, results in the high reliability of our approach, which is proved by the significant improvement in the failure rate shown in Tables 1 and 2.

Another advantage in our disk decomposition approach is the consistency of the cuts. By looking at Figure 5, we can appreciate how the algorithm produces similar decompositions at different resolutions of the same shape. Although the segmentation may be different, we can see how more detailed areas are decomposed into a larger number of small patches, while areas with lesser detail are partitioned into a few larger regions. From Figure 5 we can notice that this behavior is also consistent across different shapes. Indeed, both the octopus and the chomper have tentacles, which are segmented in a similar fashion. On the other hand, the head of the octopus is always decomposed into one or two regions, similarly to the head and the larger leaves of the chomper.

The disk decomposition also positively impacts applications where we jointly process the surface and the UV parametrization. This often happens in procedural texturing applications, where patterns emerge from surface-based simulations [MMMR22, Tur91] or numerical solutions to differential equations [Sta03, KCPS15]. In Figure 6 we show how a surface-based simulation can benefit from our disk decomposition. In particular, the presence of multiple boundaries and overlapping parametrizations results in significant discontinuities at the seam edges, as well as regions where the pattern resolution is low, despite the texture being 4096×4096 pixels (this behavior is especially visible on the legs). In contrast, our UV parametrization always produces regions that are topologically equivalent to a disk both in $\mathbb{R}^3$ and in texture space, resulting

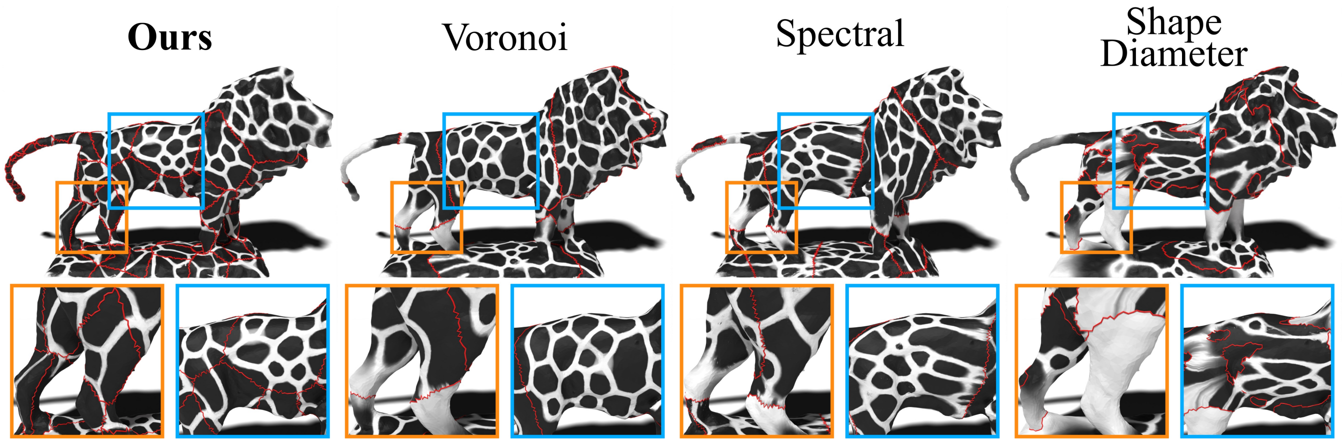

Figure 6: *Comparison between the UV parametrization produced by our approach and the other methods considered in our experiments. The resulting parametrization is used for generating a procedural texture via surface-based simulations.*

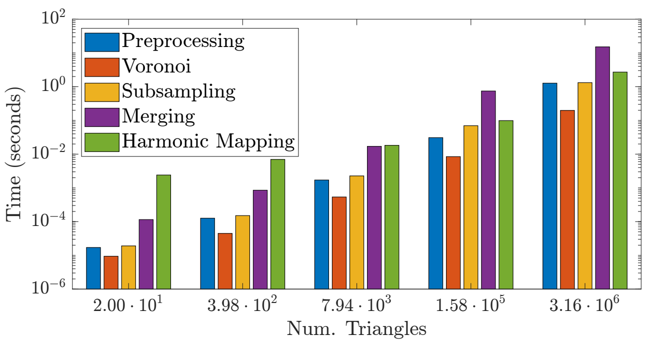

Figure 7: *Runtime of each step in our algorithm plotted against the number of triangles in the surface. Times are averaged across all the meshes with a triangle count in the corresponding order of magnitude. Both axes are plotted in logarithmic scale.*

in seamless boundary connections and a more evenly distributed texture resolution.

Finally, we present a runtime analysis of our approach in Figure 7. We partition the dataset from Shay *et al.* [SSS22] into five bins, depending on the order of magnitude of the triangle count (x-axis). We run our method, using the harmonic parametrization algorithm for the unwrapping step, and average the runtime of each stage of the pipeline (y-axis). The preprocessing step consists in the construction of the dual mesh graph. For visualization purposes, we plot the values on a log-log scale. We can observe that all the steps in the pipeline scale roughly linearly with the input size, with the merging step becoming the slowest for larger inputs. Clearly, the runtime performance of the unwrapping step depends on the choice of the algorithm, which in this case is dominant for the inputs at the lowest resolutions but scales better with as the triangle count increases.

7. Conclusions

We introduced a new method for computing a segmentation of a surface into topological disks for the UV parametrization of triangular meshes. Our fully combinatorial approach allows us to reliably parametrize surfaces of arbitrary topology and geometry with a negligible failure rate, while at the same time maintaining state-of-the-art performance in the quality of the parametrization and an efficient computational cost. We have shown that our technique generates decompositions that are consistent across different resolutions and geometries, and that it is beneficial for procedural texturing applications that require bijective UV maps.

Our multi-stage pipeline is modular by design and can be easily extended or tuned to target different and more specific applications. In particular, the parametrization of the single regions can be accomplished with any method for the unwrapping of topological disks; by processing the regions and analyzing their properties, it would also be possible to use different and more sophisticated parametrization algorithms to unwrap different regions. On top of that, the greedy approach for the merging strategy uses a simple energy that aims to minimize the perimeter-to-area ratio of the final regions; however, it would be straightforward to tune or replace that energy with a different one in order to achieve a different goal while still preserving the topological equivalence to a disk of each region. Moreover, our method can be applied as a step into more complex pipelines; for instance, one can compute an initial segmentation with certain properties and then adopt our strategy onto non-disk topologies to obtain a disk decomposition of the entire surface.

We notice that the efficiency of our implementation depends on the deployment of scalable algorithms and comes at the cost of some limitations. In particular, our approach can currently deal only with manifold meshes, as the strategy for determining the topology from the Euler characteristic is not applicable in the case of non-manifold geometries. Finally, we stress that some steps of the pipeline can be naturally improved by adopting more efficient and specific data structures (*e.g.*, structures supporting fast intersections and unions in the merging stage) or more sophisticated and advanced strategies (*e.g.*, dedicated packing algorithms for a space-efficient UV atlas).

References

[BKP*10] Botsch M., Kobbelt L., Pauly M., Alliez P., Lévy B.: *Polygon mesh processing*. CRC press, 2010. 2, 3

[Cat74] Catmull E. E.: *A subdivision algorithm for computer display of curved surfaces*. The University of Utah, 1974. 2

[CCC*08] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F., Ranzuglia G.: MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference* (2008), Scarano V., Chiara R. D., Erra U., (Eds.), The Eurographics Association. doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136. 2, 3, 5, 6

[Cra18] Crane K.: Discrete differential geometry: An applied introduction. *Notices of the AMS, Communication 1153* (2018). 2

[DC16] Do Carmo M. P.: *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016. 3

[DGWB*14] De Groot E., Wyvill B., Barthe L., Nasri A., Lalonde P.: Implicit decals: Interactive editing of repetitive patterns on surfaces. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 141–151. 2

[Dij59] Dijkstra E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271. 3

[EDD*95] Eck M., DeRose T., Duchamp T., Hoppe H., Lounsbery M., Stuetzle W.: Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 173–182. 2, 5

[EMP*02] Ebert D. S., Musgrave F. K., Peachey D., Perlin K., Worley S.: *Texturing and modeling: a procedural approach*. Elsevier, 2002. 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*, 3 (2021), 289–318. 2, 3

[GJ*10] Guennebaud G., Jacob B., et al.: Eigen v3. http://eigen.tuxfamily.org, 2010. 5

[GKK*21] Garanzha V., Kaporin I., Kudryavtseva L., Protais F., Ray N., Sokolov D.: Foldover-free maps in 50 lines of code. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–16. 2, 5

[GSC21] Gillespie M., Springborn B., Crane K.: Discrete conformal equivalence of polyhedral surfaces. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 1–20. 2, 5

[Har01] Hart J. C.: Perlin noise pixel shaders. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), pp. 87–94. 2

[Hir03] Hirani A. N.: *Discrete exterior calculus*. California Institute of Technology, 2003. 3

[JKS05] Julius D., Kraevoy V., Sheffer A.: D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum* (2005), vol. 24. 2

[JPS*13] Jacobson A., Panozzo D., Schüller C., Diamanti O., Zhou Q., Pietroni N., et al.: libigl: A simple c++ geometry processing library. *Google Scholar* (2013). 5

[KCPS15] Knöppel F., Crane K., Pinkall U., Schröder P.: Stripe patterns on surfaces. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–11. 2, 7

[KLT05] Katz S., Leifman G., Tal A.: Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8 (2005), 649–658. 2

[KT03] Katz S., Tal A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM transactions on graphics (TOG) 22*, 3 (2003), 954–961. 2

[LFXH17] Liu Y.-J., Fan D., Xu C.-X., He Y.: Constructing intrinsic delaunay triangulations from the dual of geodesic voronoi diagrams. *ACM Transactions on Graphics (TOG) 36*, 2 (2017), 1–15. 3

[LKK*18] Li M., Kaufman D. M., Kim V. G., Solomon J., Sheffer A.: Optcuts: Joint optimization of surface cuts and parameterization. *ACM transactions on graphics (TOG) 37*, 6 (2018), 1–13. 2, 6

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (2002), 10–p. 2, 5, 6

[LZ04] Liu R., Zhang H.: Segmentation of 3d meshes through spectral clustering. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.* (2004), IEEE, pp. 298–305. 2

[LZ07] Liu R., Zhang H.: Mesh segmentation via spectral embedding and contour analysis. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 385–394. 2

[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. 2, 5, 6

[MBMR22] Maggioli F., Baieri D., Melzi S., Rodolà E.: Newton's fractals on surfaces via bicomplex algebra. In *ACM SIGGRAPH 2022 Posters*. 2022, pp. 1–2. 2

[MBRM25] Maggioli F., Baieri D., Rodolà E., Melzi S.: Rematching: Low-resolution representations for scalable shape correspondence. In *Computer Vision – ECCV 2024* (Cham, 2025), Leonardis A., Ricci E., Roth S., Russakovsky O., Sattler T., Varol G., (Eds.), Springer Nature Switzerland, pp. 183–200. 3

[MM*23] Mancinelli C., Melzi S., et al.: Spectral-based segmentation for functional shape-matching. In *ITALIAN CHAPTER CONFERENCE* (2023), Eurographics Association, pp. 47–58. 2, 6

[MMM*24] Marin D., Maggioli F., Melzi S., Ohrhallinger S., Wimmer M.: Reconstructing curves from sparse samples on riemannian manifolds. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15136. 2

[MMMR22] Maggioli F., Marin R., Melzi S., Rodolà E.: Momas: Mold manifold simulation for real-time procedural texturing. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 519–527. 2, 7

[MMP87] Mitchell J. S., Mount D. M., Papadimitriou C. H.: The discrete geodesic problem. *SIAM Journal on Computing 16*, 4 (1987), 647–668. 3

[MNPP22] Mancinelli C., Nazzaro G., Pellacini F., Puppo E.: b/surf: Interactive bezier splines on surface meshes. *IEEE Transactions on Visualization and Computer Graphics 29*, 7 (2022), 3419–3435. 2

[Mor01] Morita S.: *Geometry of differential forms*, vol. 201. American Mathematical Soc., 2001. 3

[MYV93] Maillot J., Yahia H., Verroust A.: Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), pp. 27–34. 1

[NPP21] Nazzaro G., Puppo E., Pellacini F.: geotangle: interactive design of geodesic tangle patterns on surfaces. *ACM Transactions on Graphics (TOG) 41*, 2 (2021), 1–17. 2

[PSOA18] Poerner M., Suessmuth J., Ohadi D., Amann V.: adidas tape: 3-d footwear concept design. In *ACM SIGGRAPH 2018 Talks*. 2018, pp. 1–2. 1

[PTH*17] PORANNE R., TARINI M., HUBER S., PANOZZO D., SORKINE-HORNUNG O.: Autocuts: simultaneous distortion and cut optimization for uv mapping. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–11. 2

[RNP*22] RISO M., NAZZARO G., PUPPO E., JACOBSON A., ZHOU Q., PELLACINI F.: Boolsurf: Boolean operations on surfaces. *ACM Transactions on Graphics (TOG) 41*, 6 (2022), 1–13. 2

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1. 2, 5

[SH02] SHEFFER A., HART J. C.: Seamster: inconspicuous low-distortion texture seam layout. In *IEEE Visualization, 2002. VIS 2002.* (2002), IEEE, pp. 291–298. 2

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer 24*, 4 (2008), 249–259. 2, 6

[SSS22] SHAY G., SOLOMON J., STEIN O.: A dataset and benchmark for mesh parameterization. *arXiv preprint arXiv:2208.01772* (2022). 2, 5, 6, 7, 8

[Sta03] STAM J.: Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG) 22*, 3 (2003), 724–731. 2, 7

[Tur91] TURK G.: Generating textures on arbitrary surfaces using reaction-diffusion. *Acm Siggraph Computer Graphics 25*, 4 (1991), 289–298. 7

[Tut63] TUTTE W. T.: How to draw a graph. *Proceedings of the London Mathematical Society 3*, 1 (1963), 743–767. 2, 5