Diskify: a combinatorial approach to the computation of cut networks on surfaces

Filippo Maggioli[a,*], Simone Melzi[b]

[a]Department of Information Science and Technology, Pegaso University, Naples, Italy
[b]Department of Informatics, System and Communication, University of Milano-Bicocca, Milan, Italy

ARTICLE INFO

Keywords:
UV parametrization
Surface segmentation
Surface cutting

ABSTRACT

We present a reliable, fully combinatorial 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. From the disk decomposition, we propose two directions. On one side, adjacent regions are merged following a greedy approach that minimizes some objective function, all while preserving disk equivalence. Alternatively, the regions can be used to guide a cut that makes the whole surface topologically equivalent to a disk. In both cases, 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.

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 [41] and manufacturing design [51], significant efforts have been made to design recursive or repetitive patterns [48, 6], curves [43, 44], and shapes [55] 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 [61, 30, 39], or baked with solid textures [10, 23, 37].

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 [59], making robustness and predictability critical requirements for practical use.

*Corresponding author
maggioli.filippo@gmail.com (F. Maggioli);
simone.melzi@unimib.it (S. Melzi)
ORCID(s): 0000-0001-8008-8468 (F. Maggioli);
0000-0003-2790-9591 (S. Melzi)

This paper extends [40], where we introduced an 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 was 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 [40] was to minimize the number of failure cases, which are a persistent obstacle in existing UV mapping pipelines. With this work, we significantly expand our previous contribution by demonstrating that the segmentation algorithm and the merging approach are not metric-dependent and can be seen as a general pipeline that can be easily adapted to the desired task. For instance, in our previous contribution we only considered the standard geodesic distance to produce regions with minimal perimeter-to-area ratio; in this extended work, we propose a discrete version of the isophotic metric [53], which allows us to localize region boundaries at sharp features.

Furthermore, we consider a different scenario in which segmentation is not a suitable solution and, instead, a disk development of the surface is required. Cutting a surface open and making it topologically equivalent to a disk is a classical problem in geometry and topology [47], and has applications for surface parametrization [21] and fabrication [15]. However, the problem of finding optimal cuts is

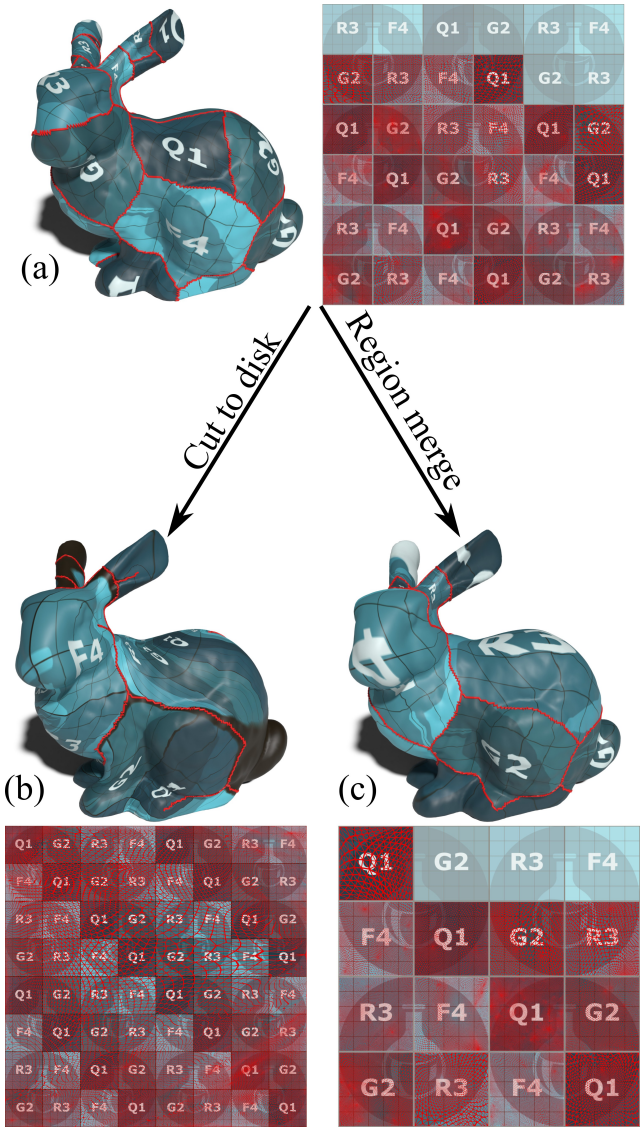

Figure 1: A mesh segmented into regions with disk topology (a) can be easily parametrized. Those regions can be then connected to build a cut that makes the surface homeomorphic to a disk (b) or merged to reduce the number of regions with a regular shape (c).

NP-hard [12], and even existing approximation can involve costly optimizations processes [21] or solving challenging geometrical problems [49]. By exploiting the adjacency graph in the segmentation, we provide a fully combinatorial, mathematically correct algorithm to identify a surface cut that produces disk topology and can be tuned according to the required metric (see Figure 1). While our approach does not provide optimality guarantees, the experiments show that results are in line with other, more costly, solutions based on optimization processes, and that our cuts are significantly shorter than those manually introduced by artists (see Table 2 in Section 5).

In summary, with this work we provide a general, efficient, and reliable pipeline for the identification of cut networks on arbitrary surfaces. Our approach is fully combinatorial and produces cuts that can either be used to segment the mesh or to open it into a topological disk. Furthermore, our algorithms come with provable mathematical guarantees, and can be tuned with a variety of metrics to achieve different segmentation and parametrization goals.

We organize the remainder of this paper 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 analyze the robustness of the proposed method to varying mesh densities, 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 [2] and makes it one of the most active research areas in geometry processing, even in recent years [1, 5]. Consequently, a wide variety of different approaches have been developed to accomplish the task of computing a UV parametrization of a surface [17].

The standard pipeline to produce a UV parametrization consists of computing cuts of the surface and then parametrizing regions separately [3], 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 [52, 32]. 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. In our previous work [40], we proposed a robust algorithm to segment the surface into easily parametrizable topological disks, but the method was constrained by a specific metric and was applicable only to cases where multiple UV islands are acceptable.

UV parametrization. Primarily, studies and efforts have focused on computing a piecewise linear map from a surface to the real plane, without concern for segmenting or cutting the surface. The Tutte embedding [63] 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 [25]. Well-established examples are: (i) the harmonic mapping, which minimizes distortion and metric dispersion [11], (ii) the conformal mapping, which locally preserves the angles [31], and (iii) the as-rigid-as-possible (ARAP) parametrization introduced by Liu *et al.* [33], 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) [54], which extends the

ARAP metric with flip-free energies. Garanzha *et al.* [18] 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.* [19], 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 [28] or regions with high curvature [60]. For a more detailed survey on the early strategies of mesh segmentation, we refer to [56]. Other common strategies were based on the Voronoi decomposition from a sparse sample set [3], but also to deploy hierarchical structures [29], and the introduction of developability energies based on Gaussian curvature [27]. Shapira *et al.* [57] 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 [34] and contour analysis [35] in the spectral embedding. More recently, Mancinelli *et al.* [42] 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.

Disk development. Rather than segmenting the surface, it is well-known that, given an arbitrary surface of genus g, it is always possible to find a cut network that opens the surface into a topological disk [47]. A particularly interesting case in geometry and topology is the cut network of minimum length, whose computation has been proved to be NP-hard [12]. Earlier approaches constructed the cuts by expanding from a single point [14, 16] or exploited user interactivity to supervise the choice of the cut [50]. Later, Erickson *et al.* investigated greedy approaches for approximating the minimum cut network in polynomial time using spanning trees to connect boundary components [12] or to identify sub-graphs and cycles in the homology basis of the surface [13]. However, these approaches focus on finding the minimum number of cuts, rather than a cut of minimal length, while other related results outsource the homology computation, which is a non-trivial task in the general case [49]. A successful approach for the identification of a cut network for disk development has been introduced by Gu *et al.* with the *Geometry Images* work [21]. Similarly to modern UV parametrization techniques, this method jointly computes the cut and the parametrization into the unit square through an iterated optimization process, which is based on a previous solution from Dey *et al.* [7]. While effective on arbitrary surfaces, this approach specifically optimizes for cuts and parametrizations that allow for seamless blending of surface coordinates and normal vectors. Consequently, the resulting cuts are not best suited for minimal length constraints or applications where localization (*e.g.*, at sharp features) is required.

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 Riemmanian manifold, we remand to [9, 46].

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.* [17] 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;
- $T \subset V^3$ is a set of triangles between vertices in V;
- $E \subset V^2$ is a set of edges inferred from the triangles.

Although we admit meshes with self-intersections and an arbitrary number of boundary loops, 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).

Appendix D discusses how our solution can be extended to non-manifold meshes.

Given a triangular mesh $M = (V, E, T)$, as discussed in [24], 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 [24].

A distance function over the surface is discretized as a function $d_M : V \times V \to \mathbb{R}$ that satisfies the metric axioms w.r.t. the space of mesh vertices [20, §1]. While the exact geodesic distance can also be defined for triangular meshes [45], also shortest paths on the mesh graph are a perfectly acceptable distance function, often preferable to the exact geodesic distance due to the higher efficiency and numerical stability of Dijkstra's algorithm [8].

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.* [1] 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 as follows:

1. produce a Voronoi decomposition of the surface according to the given metric (Section 4.1);
2. refine the decomposition to ensure regions have disk topology (Section 4.2.1), then choose:
   (a) merge regions to optimize objective function while keeping the disk topology (Section 4.2.2);
   (b) use regions boundaries to identify a cut that develops the surface into a disk (Section 4.2.3);
3. compute the UV map of each region or produce a UV parametrization for the entire surface (Section 4.3).

4.1. Voronoi decomposition

As discussed in Section 3, a segmentation of a triangular mesh consists of 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, also used by well-known geometry processing software such as MeshLab [3], for partitioning discrete surfaces with respect to their vertices 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 $\mathrm{Vor}(\hat{M}, P) = \{\mathrm{Vor}(\hat{M}, p_1), \cdots, \mathrm{Vor}(\hat{M}, p_n)\}$ such that

$$\mathrm{Vor}(\hat{M}, p_i) = \left\{ v \in \hat{M} \;:\; p_i = \operatorname*{argmin}_{p_j \in P} d_{\hat{M}}(v, p_j) \right\}, \quad (2)$$

according to some distance function $d_{\hat{M}}$.

Algorithm 1 Subsampling algorithm.

```
procedure SUBSAMPLING(M̂, P, k)
    S ← ∅
    for p_i ∈ P do
        M_i ← submesh induced by Vor(M̂, p_i)
        if χ(M_i) = 1 then
            S ← S ∪ {M_i}
        else
            P' ← VORONOIFPS(M_i, k)
            S_i ← SUBSAMPLING(M_i, P', k)
            S ← S ∪ S_i
        end if
    end for
    return S
end procedure
```

To ensure an even coverage of the surface and an efficient decomposition, we employ the algorithm presented by Maggioli *et al.* [38], that jointly computes a farthest point sampling of n sample points and their Voronoi decomposition in time $\mathcal{O}\left(|\hat{V}| \log |\hat{V}| \log n\right)$. While this method requires applying Dijkstra's algorithm to the dual mesh graph, we stress that the edge weights can be defined using arbitrary distance functions.

In our experiments, we test our algorithm using two different distance functions for weighting the edges: the geodesic distance between the barycenters of adjacent triangles, and the dihedral angle between the triangles. Details are provided in Section B.

4.2. Decomposition refinement

Given an initial Voronoi decomposition of the surface, we then consider a refinement strategy that produces either a decomposition of the mesh into topological disks (Section 4.2.2) or a cut network along the edges that develops the mesh into a topological disk (Section 4.2.3).

4.2.1. 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.* [36] 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

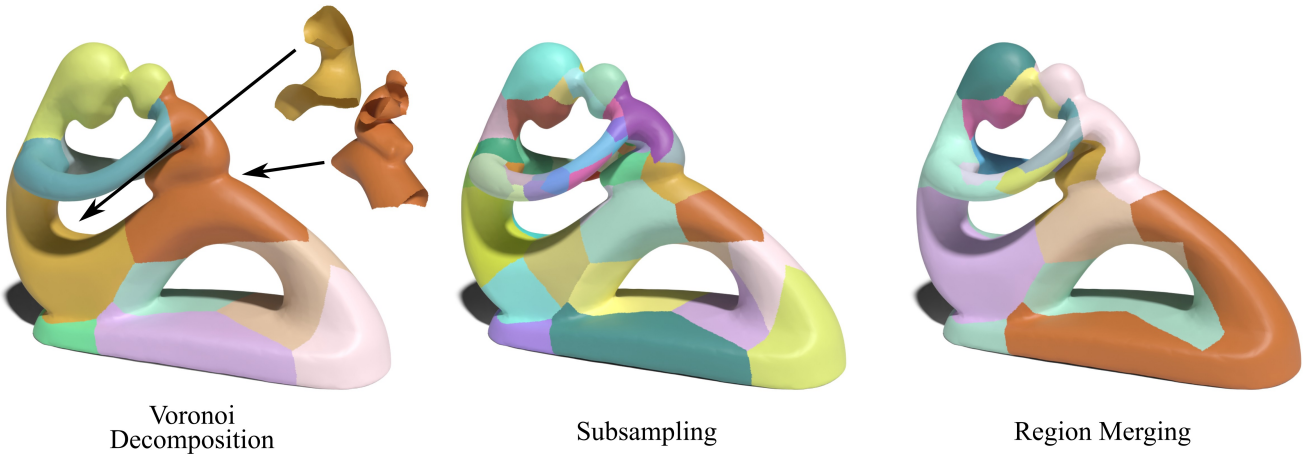

Figure 2: Left: the geodesic Voronoi decomposition of the surface produces regions that are not topologically equivalent to a disk. Middle: our first refinement step adds sample points to the critical areas, ensuring that each Voronoi region is a topological disk. Right: our greedy merging algorithm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

may already satisfy this condition. Consequently, we only need to further decompose the other regions. Consider the sample p_i and the Voronoi region $\mathrm{Vor}(\hat{M}, p_i)$ 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 of the mesh into topological disks (see the middle frame of Figure 2). The complexity of the algorithm is $\mathcal{O}\left(|V| \log^2 |V| \log k\right)$, and the analysis is reported in Appendix C.

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. See Appendix A. □

4.2.2. Region merging

As shown in the middle plot of Figure 2, the subsampling procedure could produce a large number 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 to iteratively merge adjacent regions while maintaining a decomposition into topological disks. Our approach takes advantage of the inclusion–exclusion principle for the Euler characteristic with compact support in a locally compact space, that is

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

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.

Algorithm 2 Region merging algorithm.

```
procedure RegionMerging(M, S, ε)
    G = (V, E) ← dual graph of segmentation S
    while G is unchanged do
        Q ← empty priority queue
        for (S_i, S_j) ∈ E do
            if χ(S_i ∩ S_j) = 1 then
                ϑ_ij ← Score(M, S_i, S_j)
                if ϑ_ij ≥ ε then
                    Q.push(ϑ_ij, (S_i, S_j))
                end if
            end if
        end for
        if |Q| > 0 then
            (S_i, S_j) ← Q.pop()
            Merge S_i with S_j and update G
        end if
    end while
end procedure
```

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 then filter out all edges between regions whose union does not form a topological disk, and for the remaining edges, we compute a score that depends solely on the two regions and can be defined by the application. We also consider using a threshold ε for the score to exclude all pairs of regions that would merge into a

Algorithm 3 Cut identification algorithm

1: **procedure** CUTTODISK(M, S, w: weight function)
2: $\mathcal{E} \leftarrow \emptyset$
3: **for** $i \neq j$ **do**
4: **for** $\mathcal{B} \in S_i \cap S_j$ **do**
5: **if** $\mathcal{B}$ is not a point **then**
6: $\mathcal{E} \leftarrow \mathcal{E} \cup \{(S_i, S_j, \mathcal{B}, w(\mathcal{B})\}$
7: **end if**
8: **end for**
9: **end for**
10: $\mathcal{G} \leftarrow (S, \mathcal{E})$
11: $\mathcal{T} \leftarrow$ MINSPANTREE($\mathcal{G}$)
12: $\mathcal{F} \leftarrow \mathcal{E} \setminus \mathcal{T}$
13: $\mathcal{B}_{\mathcal{F}} \leftarrow \emptyset$
14: **for** $e = (S_i, S_j, \mathcal{B}_e, w(\mathcal{B}_e)) \in \mathcal{F}$ **do**
15: $\mathcal{B}_{\mathcal{F}} \leftarrow \mathcal{B}_{\mathcal{F}} \cup \mathcal{B}_e$
16: **end for**
17: Cut M along $\mathcal{B}_{\mathcal{F}}$
18: **end procedure**

region that does not satisfy the quality criterion captured by the score. If there are edges left, we select the edge with the highest score, collapse the nodes in the graph, and merge the corresponding regions. This whole process iterates until the graph remains unchanged, indicating that no more regions can be merged. In Appendix C, we prove that the time complexity of Algorithm 2 is $\mathcal{O}\left(|S|^2|V|\right)$.

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

4.2.3. Cut identification

Alternatively to the region merging step, we consider the possibility of identifying a subset of the region boundaries realizing a cut that develops the surface into a topological disk. In this regard, we expand the idea of dual graph for a segmentation S and introduce the notion of *dual multi-graph*. Similarly to a dual graph, the vertices of the multi-graph correspond to the regions in the segmentation, but in this case adjacent vertices are allowed to be connected through multiple edges. In particular, each edge connecting S_i and S_j corresponds to a connected component in the boundary $S_i \cap S_j$. In order to avoid ill-defined cuts, we ignore all the boundary components corresponding to single points.

Theorem 1. *Let S be a segmentation of a connected surface $\mathcal{M}$ into topological disks, such that $|S| \geq 3$, and let $\mathcal{G}$ be the dual multi-graph induced by S. Let us consider a spanning tree $\mathcal{T}$ for the multi-graph $\mathcal{G}$, and consider the set $\mathcal{F} = \mathcal{E} \setminus \mathcal{T}$ of edges that are not in $\mathcal{T}$. The boundary components associated to the edges in $\mathcal{F}$ form a curves network $\mathcal{B}_{\mathcal{F}}$ that cuts $\mathcal{M}$ into a topological disk.*

Proof. See Appendix A. □

The cut identification algorithm, detailed in Algorithm 3, describes the procedure we use for cutting surfaces to a disk. Lines 2-10 describe the process for building the dual multi-graph from the segmentation. Namely, for each pair of adjacent regions S_i and S_j, we iterate over the connected components of their boundary. If the component $\mathcal{B}$ is not a single point, we add an edge between S_i and S_j, associated with $\mathcal{B}$ and weighted according to an input weight function. By Theorem 1, the complementary edges to the minimum spanning tree (Lines 11-12) identify a cut of the surface into a topological disk. The remainder of the algorithm (Lines 13-17) consists in computing the curves on the surfaces associated to cut edges, and executing the surface cut. The time complexity of the algorithm is $\mathcal{O}(|V| \log |V|)$, and a detailed analysis is provided in Appendix C.

4.3. Parametrization

The major advantage of our approach, is that it is not bounded to a specific UV parametrization algorithm. Indeed, our method cuts the surface into one or more topological disks, which are easier to parametrize using well-know methods like harmonic mapping [11], conformal mapping [31], and as-rigid-as-possible mapping [33]. Consequently, our method can be plugged in any parametrization pipeline relying on a segmentation of the surface [63, 54, 18], but it can also be injected into algorithms that jointly perform segmentation and parametrization [19].

When the surface is segmented into multiple disks, 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. We note that the grid arrangement is only relative to the placement of the regions; in case of boundary-free parametrization methods, we don't constrain the boundary and only apply a rescaling to make the region fit inside the grid cell. Although this approach does not produce an optimal packing, it is very efficient and can be easily replaced with more optimized algorithms.

5. Results

We implemented our method in C++, using Eigen [22] and libigl [26]. The source code is available at `https://github.com/filthynobleman/comb-uv`.

Baselines. For the UV parametrization step, we consider the harmonic mapping algorithm proposed by Eck *et al.* [11], eventually refined by the as-rigid-as-possible parametrization introduced by Liu *et al.* [33], as well as the conformal mapping technique presented by Lèvy *et al.* [31]. For all of these algorithms, we rely on the implementation provided in libigl [26]. Using these parametrization techniques, we compare our segmentation approach (Algorithms 1 and 2) against various segmentation techniques. As a naive baseline, we consider a geodesic Voronoi segmentation (Voronoi) [3], plus an iterated version that guarantees

	segmentation	area dist.	angle dist.	flipped triangles	cut length	failure rate
Harmonic	Voronoi	0.75	1.39	1.03%	13.53	3.15%
	Voronoi Iter.	0.72	1.39	0.43%	13.14	0.4%
	Shape Diam.	0.85	1.44	3.45%	0.43	50.99%
	Spectral	0.99	1.98	2.07%	11.54	58.28%
	Disks (Geod.)	0.87	1.08	1.47%	3.11	0.04%
	Disks (Angle)	0.94	1.36	0.17%	23.64	1.62%
Conformal	Voronoi	0.42	0.27	2.56%	7.58	87.26%
	Voronoi Iter.	0.80	5.22	0.75%	12.76	8.37%
	Shape Diam.	0.63	0.96	10.41%	0.33	68.29%
	Spectral	0.79	0.77	7.18%	6.11	85.61%
	Disks (Geod.)	0.78	0.56	4.90%	1.78	44.38%
	Disks (Angle)	1.03	5.99	0.32%	13.64	7.56%
ARAP	Voronoi	0.45	0.36	2.62%	13.53	3.16%
	Voronoi Iter.	0.91	4.31	16.26%	12.67	4.75%
	Shape Diam.	0.43	1.01	9.26%	0.40	51.27%
	Spectral	0.49	0.51	3.04%	11.54	58.28%
	Disks (Geod.)	0.59	0.77	5.62%	3.11	0.09%
	Disks (Angle)	1.15	4.27	7.88%	23.74	2.08%
	Artist	0.28	0.91	7.45%	5.52	0%
	OptCuts	0.05	0.13	~0%	1.47	12.09%

Table 1
Results on the parametrization benchmark [59] 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	Voronoi	0.83	1.48	1.47%	16.15	0.32%
	Voronoi Iter.	0.81	1.47	0.55%	17.43	0.37%
	Shape Diam.	1.25	1.92	6.52%	0.83	68.63%
	Spectral	0.99	1.98	2.07%	11.54	58.28%
	Disks (Geod.)	0.99	1.22	1.94%	4.63	0%
	Disks (Angle)	1.01	1.41	0.38%	14.59	0%
Conformal	Voronoi	0.45	0.35	3.42%	7.77	84.69%
	Voronoi Iter.	0.85	5.28	0.95%	16.88	9.12%
	Shape Diam.	0.98	1.67	2.23%	0.81	82.25%
	Spectral	0.79	0.77	7.18%	6.11	85.61%
	Disks (Geod.)	0.96	0.71	6.34%	2.90	50.01%
	Disks (Angle)	1.07	6.28	4.91%	8.92	6.28%
ARAP	Voronoi	0.47	0.44	3.72%	16.15	0.34%
	Voronoi Iter.	0.97	4.68	15.19%	16.85	0.59%
	Shape Diam.	0.65	1.46	15.12%	0.75	68.84%
	Spectral	0.49	0.51	3.04%	11.54	58.28%
	Disks (Geod.)	0.68	0.90	7.06%	4.63	0.02%
	Disks (Angle)	1.22	4.92	10.01%	14.83	1.79%
	Artist	0.34	1.21	10.22%	7.48	0%
	OptCuts	0.05	0.15	~0%	2.12	17.27%
	Cut (Geod.)	1.07	2.30	1.79%	4.09	0%
	Cut (Angle)	1.06	2.17	2.13%	4.45	0%

Table 2
Results on the parametrization benchmark [59] 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.

disk topology (Voronoi Iter.) [11]. The shape diameter segmentation proposed by Shapira *et al.* (Shape Diameter) [57] is a standard clustering algorithm that produces short seams localized at sharp features. We also consider a spectral method introduced by Mancinelli *et al.* (Spectral) [42] for segmenting surfaces into semantically distinct regions. For reference, we report the benchmark evaluation on the cuts and parametrizations provided by the artists, as well as the results of the OptCuts algorithm [32], which jointly optimize for a segmentation and a UV parametrization of the surface. For non-disk input meshes, we also consider the parametrization obtained by cutting the mesh into a disk with Algorithm 3 and mapping it with the Tutte's embedding [63]. For our methods, we consider two variants using different distance metrics, score functions, and boundary weights. Details are reported in Appendix B. We use an initial sampling with 10 samples and a recursive refinement with 5 samples. However, we provide in Appendix D a discussion on how to automate the number of samples.

Dataset. We conduct our experiments on the parametrization benchmark proposed by Shay *et al.* [59]. 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). Since we focus on segmentation algorithms, we select the manifold meshes of the *Uncut* category.

Metrics. Each shape in [59] 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. The benchmark application compares the 3D triangles and their parametrization in 2D, and reports the average relative difference between their areas (*area distortion*) and their angles (*angle distortion*). It also computes the average length of the cut over the surface and the average percentage of flipped triangles, where a triangle is considered flipped if its Jacobian has non-positive determinant. If the segmentation (or the parametrization) algorithm crashes, or if the parametrization has null, infinite, or undefined area, the benchmark considers it as a failure case. 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.

Evaluation. We summarize the results of our comparisons in Table 1, averaging all the metrics on the cases for which the methods successfully find a parametrization. The percentage of failure cases is shown in the last column. We stress that, as it is standard practice in texture mapping, we parametrize regions to the unit square, which can cause the UV map of some boundary facets to be degenerate and cause a non-negligible amount of flips even with robust mapping algorithms such as the Harmonic embedding. 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 shape diameter approach and the spectral based method produce segmentations that cannot be parametrized in a significant number of cases. This is mainly due to these method being unable to guarantee disk topology, often producing non flattenable or non convex regions. This is even more evident when we compare the

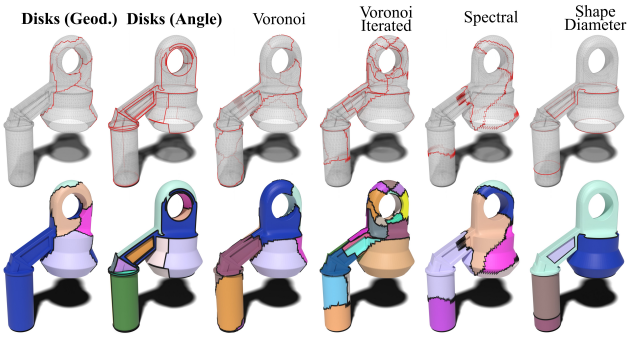

Figure 3: Comparison between the cuts along the mesh introduced by the different segmentation algorithms on an object in the parametrization benchmark dataset from Shay *et al.* [59]. Top row: objects are rendered in slight transparency to show the cuts on the other side. Bottom row: regions are colored differently to highlight the segmentation.

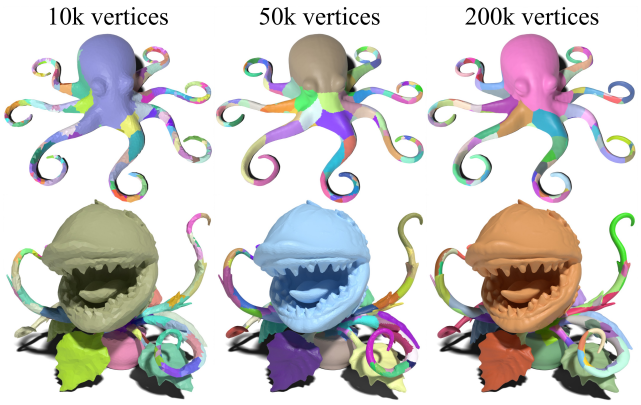

Figure 4: Our disk segmentation algorithm applied to meshes at different resolutions, using the same set of starting Voronoi samples.

reliability of the base Voronoi decomposition with its iterated version that guarantees disk topology, highlighting how a disk decomposition can be beneficial to parametrization tasks. However, as can be noticed in Figure 3, the total absence of geometric priors and refinement strategies in the iterative Voronoi decomposition tends to produce very fine segmentations with a long total cut length. 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 significantly less and that meshes without disk topology are more challenging; this suggests that disk decomposition simplifies the parametrization by favoring the convergence of the eigensolver, an hypothesis supported by the results obtained with the iterative Voronoi decomposition. In particular, with the angle-based distance, the failure rate drops drastically, suggesting that this metrics tends to produce flatter regions and induces a more suitable decomposition for conformal parametrization. This behavior is also confirmed by the reduced number of flipped triangles. However, such metric tends to produce irregular regions and unreasonably long cuts (see Figure 3), negatively affecting the area and angle distortion in the map; similar results are obtained with the iterative Voronoi decomposition, which also produces very small regions with irregular boundaries.

In contrast, the geodesic metric seems to be more suitable to improve the regions regularity and reduce the total length of cuts. Due to the region merging score that minimizes the perimeter-to-area ratio, our method is able to drastically reduce the amount and the total length of cuts needed by the simple and iterated Voronoi decompositions, highlighting the actual contribution of our pipeline. Similarly, when the spectral-based method 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 cut length (see Figure 3). From Figure 3, we can also appreciate that the shape diameter approach tends to produce clean cuts, aligned with sharp features of the object. This results 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, resulting 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 step.

For most of the approaches derived from our contribution, the results are aligned with the parametrization obtained by the artists. This suggests that our method can be employed as a fast and reliable parametrization technique, especially in applications where no fine control over seam placement is needed (*i.e.*, procedural texture baking).

Finally, we see that the OptCuts [32] method achieves impressive results in terms of parametrization quality, but this comes at the cost of a non-negligible failure rate due to numerical errors in the optimization procedure, which result in diverging or undefined texture coordinates. On top of that, we stress that OptCuts assumes the possibility of remeshing the surface prior to the optimization procedure, making it unsuitable for many practical applications.

6. Analysis

A major advantage of both our approaches (*i.e.*, shape segmentation and surface cut) is that they are fully combinatorial. By not involving numerical steps in our procedure, we are always able to produce a valid decomposition of the surface or a valid cut, independently of the geometry and connectivity of the mesh. This, combined with the fact that the parametrization algorithms can be fed with only disks, results in the high reliability of our approaches, which is proved by the significant improvement in the failure rate shown in Tables 1 and 2.

Another advantage of our disk decomposition approach is the consistency of the cuts. By looking at Figure 4, we can

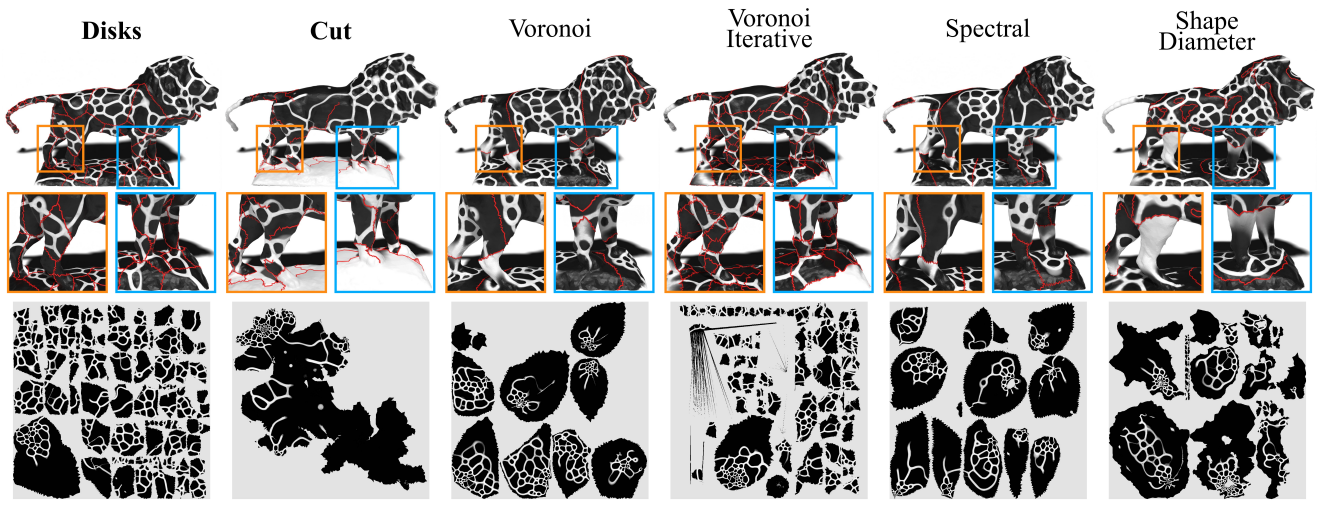

Figure 5: Comparison between the UV parametrization produced by our approaches and the other methods considered in our experiments. The resulting parametrization is used for generating a procedural texture via surface-based simulations. The texture image is shown on the bottom row (gray is used to identify unused areas of the texture).

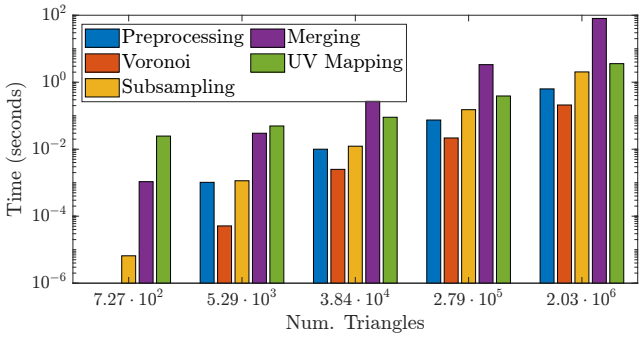

Figure 6: Runtime of each step in our disk segmentation algorithm plotted against the number of triangles in the surface. Times are averaged across all the meshes with a triangle count in the corresponding order of magnitude. Both axes are plotted in logarithmic scale.

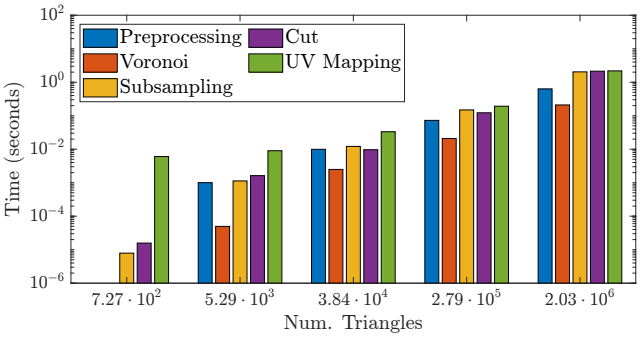

Figure 7: Runtime of each step in our surface cut algorithm plotted against the number of triangles in the surface. Times are averaged across all the meshes with a triangle count in the corresponding order of magnitude. Both axes are plotted in logarithmic scale.

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 4, 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 [39, 62] or numerical solutions to differential equations [61, 30]. In Figure 5 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 dramatic resolution inconsistencies even between adjacent regions, despite the texture being 4096×4096 pixels (this behavior is especially visible on the legs). In contrast, our decomposition algorithm always produces regions that are topologically equivalent to a disk both in $\mathbb{R}^3$ and in texture space, resulting in seamless boundary connections and a more evenly distributed texture resolution. While, in principle, this is true for the parametrization obtained from the cut surface, the cut does not provide any guarantees about the map distortion, resulting in dramatic stretches that can negatively affect the quality of the texture. Similarly, the analysis of the iterative Voronoi decomposition shows that the straightforward decomposition into disks is not sufficient. Without a refinement strategy, poorly triangulated regions (*e.g.*, the bottom of the base) could produce regions with highly irregular shapes; as a result, those regions and their immediate surrounding suffer from the same type of artifacts as the other decomposition approaches.

For all decomposition, we used the conformal mapping and texture packing algorithm implemented in Blender 5.1 [4].

Finally, we present a runtime analysis of our approach in Figures 6 and 7. We partition the dataset from Shay *et al.* [59] into five bins, depending on the order of magnitude of the triangle count (x-axis). We run our method, using the Tutte's parametrization algorithm [63] 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. Remarkably, we notice that computing the cut is significantly more efficient than merging regions. Computing the cut requires a single iteration over the boundaries of the regions: when the multi-graph is computed, we only have to compute the minimum spanning tree (see Algorithm 3). Instead, the region merging procedure discussed in Algorithm 2 re-evaluates the boundaries at every iteration, as merging two regions could potentially change their relationship with other nearby regions. 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 as the triangle count increases.

7. Conclusions

We expanded our method (originally introduced in [40]) for computing a segmentation of a surface into topological disks, generalizing the segmentation algorithm to be guided by any locally defined metric. We further extended our solution to exploit the disk segmentation and compute a cut of the surface that develops it into a topological disk. Our approach is fully combinatorial, allowing us to reliably compute cuts and segmentations for surfaces of arbitrary topology and geometry with negligible failure rate. Our solutions have provably low computational complexity, and the provided implementation is practically efficient in terms of runtime, making it a robust and scalable tool for different downstream applications. The produced cut networks can be reliably used to compute a fast parametrization of the surface, which is beneficial for procedural texturing applications requiring bijective UV maps. When used for parametrization purposes, our disk decomposition strategy produces state-of-the-art UV maps, comparable with handcrafted artist works. Additionally, our solution for disk development can efficiently compute cuts of short length, providing a useful tool for fast approximation of optimal cuts.

Our multi-stage pipeline is modular by design and can be easily extended or tuned to target different and more specific applications, as demonstrated by our use of different metrics. 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 more sophisticated parametrization algorithms to unwrap different regions. Our method can also be applied as a step into more complex pipelines; for instance, one can compute an initial segmentation with properties required for a specific application 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. Specifically, we employ a fast Voronoi decomposition algorithm to approximate distances with Dijkstra's shortest paths relative to the provided metric. Consequently, our solution can only be applied on locally defined metrics and the Voronoi tessellation is not required to be centroidal. 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

[1] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., 2010. Polygon mesh processing. CRC press.

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

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

[4] Community, B.O., . Blender - a 3D modelling and rendering package. Blender Foundation. Stichting Blender Foundation, Amsterdam. URL: http://www.blender.org.

[5] Crane, K., 2018. Discrete differential geometry: An applied introduction. Notices of the AMS, Communication 1153.

[6] De Groot, E., Wyvill, B., Barthe, L., Nasri, A., Lalonde, P., 2014. Implicit decals: Interactive editing of repetitive patterns on surfaces, in: Computer Graphics Forum, Wiley Online Library. pp. 141–151.

[7] Dey, T.K., 1994. A new technique to compute polygonal schema for 2-manifolds with application to null-homotopy detection, in: Proceedings of the tenth annual symposium on Computational geometry, pp. 277–284.

[8] Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.

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

[10] Ebert, D.S., Musgrave, F.K., Peachey, D., Perlin, K., Worley, S., 2002. Texturing and modeling: a procedural approach. Elsevier.

[11] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W., 1995. Multiresolution analysis of arbitrary meshes, in: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 173–182.

[12] Erickson, J., Har-Peled, S., 2002. Optimally cutting a surface into a disk, in: Proceedings of the eighteenth annual symposium on Computational geometry, pp. 244–253.

[13] Erickson, J., Whittlesey, K., 2005. Greedy optimal homotopy and homology generators, in: SODA, pp. 1038–1046.

[14] Farin, G., 2014. Curves and surfaces for computer-aided geometric design: a practical guide. Elsevier.

[15] Ferguson, H., Rockwood, A., Cox, J., 1992. Topological design of sculptured surfaces, in: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 149–156.

[16] Firby, P., Gardiner, C., 2001. Surface Topology. Ellis Horwood series in mathematics and its applications, Woodhead Publishing. URL: https://books.google.it/books?id=Wv6jAgAAQBAJ.

[17] Fu, X.M., Su, J.P., Zhao, Z.Y., Fang, Q., Ye, C., Liu, L., 2021. Inversion-free geometric mapping construction: A survey. Computational Visual Media 7, 289–318.

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

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

[20] Gromov, M., 2007. Metric structures for Riemannian and non-Riemannian spaces. Springer.

[21] Gu, X., Gortler, S.J., Hoppe, H., 2002. Geometry images, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 355–361.

[22] Guennebaud, G., Jacob, B., et al., 2010. Eigen v3. http://eigen.tuxfamily.org.

[23] Hart, J.C., 2001. Perlin noise pixel shaders, in: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, pp. 87–94.

[24] Hirani, A.N., 2003. Discrete exterior calculus. California Institute of Technology.

[25] Hormann, K., Lévy, B., Sheffer, A., 2007. Mesh parameterization: Theory and practice, in: ACM SIGGRAPH 2007 Courses-International Conference on Computer Graphics and Interactive Techniques.

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

[27] Julius, D., Kraevoy, V., Sheffer, A., 2005. D-charts: Quasi-developable mesh segmentation., in: Computer Graphics Forum.

[28] Katz, S., Leifman, G., Tal, A., 2005. Mesh segmentation using feature point and core extraction. The Visual Computer 21, 649–658.

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

[30] Knöppel, F., Crane, K., Pinkall, U., Schröder, P., 2015. Stripe patterns on surfaces. ACM Transactions on Graphics (TOG) 34, 1–11.

[31] Lévy, B., Petitjean, S., Ray, N., Maillot, J., 2002. Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics 21, 10–p.

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

[33] Liu, L., Zhang, L., Xu, Y., Gotsman, C., Gortler, S.J., 2008. A local/global approach to mesh parameterization, in: Computer graphics forum, Wiley Online Library. pp. 1495–1504.

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

[35] Liu, R., Zhang, H., 2007. Mesh segmentation via spectral embedding and contour analysis, in: Computer Graphics Forum, Wiley Online Library. pp. 385–394.

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

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

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

[39] Maggioli, F., Marin, R., Melzi, S., Rodolà, E., 2022b. Momas: Mold manifold simulation for real-time procedural texturing, in: Computer Graphics Forum, Wiley Online Library. pp. 519–527.

[40] Maggioli, F., Melzi, S., 2025. Uv parametrization via topological disk segmentation of surfaces, in: Eurographics Italian Chapter Proceedings - Smart Tools and Applications in Graphics, STAG. doi:10.2312/stag.20251323.

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

[42] Mancinelli, C., Melzi, S., et al., 2023. Spectral-based segmentation for functional shape-matching, in: ITALIAN CHAPTER CONFERENCE, Eurographics Association. pp. 47–58.

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

[44] Marin, D., Maggioli, F., Melzi, S., Ohrhallinger, S., Wimmer, M., 2024. Reconstructing curves from sparse samples on riemannian manifolds, in: Computer Graphics Forum, Wiley Online Library. p. e15136.

[45] Mitchell, J.S., Mount, D.M., Papadimitriou, C.H., 1987. The discrete geodesic problem. SIAM Journal on Computing 16, 647–668.

[46] Morita, S., 2001. Geometry of differential forms. volume 201. American Mathematical Soc.

[47] Munkres, J., 2000. Topology. Pearson modern classic, Pearson Education International. URL: https://books.google.it/books?id=OogGyAEACAAJ.

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

[49] Pellikka, M., Suuriniemi, S., Kettunen, L., Geuzaine, C., 2013. Homology and cohomology computation in finite element modeling. SIAM Journal on Scientific Computing 35, B1195–B1214.

[50] Piponi, D., Borshukov, G., 2000. Seamless texture mapping of subdivision surfaces by model pelting and texture blending, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 471–478.

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

[52] Poranne, R., Tarini, M., Huber, S., Panozzo, D., Sorkine-Hornung, O., 2017. Autocuts: simultaneous distortion and cut optimization for uv mapping. ACM Transactions on Graphics (TOG) 36, 1–11.

[53] Pottmann, H., Steiner, T., Hofer, M., Haider, C., Hanbury, A., 2004. The isophotic metric and its application to feature sensitive morphology on surfaces, in: European Conference on Computer Vision, Springer. pp. 560–572.

[54] Rabinovich, M., Poranne, R., Panozzo, D., Sorkine-Hornung, O., 2017. Scalable locally injective mappings. ACM Transactions on Graphics (TOG) 36, 1.

[55] Riso, M., Nazzaro, G., Puppo, E., Jacobson, A., Zhou, Q., Pellacini, F., 2022. Boolsurf: Boolean operations on surfaces. ACM Transactions on Graphics (TOG) 41, 1–13.

[56] Shamir, A., 2008. A survey on mesh segmentation techniques, in: Computer graphics forum, Wiley Online Library. pp. 1539–1556.

[57] Shapira, L., Shamir, A., Cohen-Or, D., 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. The Visual Computer 24, 249–259.

[58] Sharp, N., 2021. Intrinsic Triangulations in Geometry Processing. Ph.D. thesis. Carnegie Mellon University, USA.

[59] Shay, G., Solomon, J., Stein, O., 2022. A dataset and benchmark for mesh parameterization. arXiv preprint arXiv:2208.01772 .

[60] Sheffer, A., Hart, J.C., 2002. Seamster: inconspicuous low-distortion texture seam layout, in: IEEE Visualization, 2002. VIS 2002., IEEE. pp. 291–298.

[61] Stam, J., 2003. Flows on surfaces of arbitrary topology. ACM Transactions On Graphics (TOG) 22, 724–731.

[62] Turk, G., 1991. Generating textures on arbitrary surfaces using reaction-diffusion. Acm Siggraph Computer Graphics 25, 289–298.

[63] Tutte, W.T., 1963. How to draw a graph. Proceedings of the London Mathematical Society 3, 743–767.

A. Proofs

Proposition 2. *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. □

Lemma 1. *Let S be a segmentation of a connected surface $\mathcal{M}$ into topological disks, such that $|S| \geq 3$. For each $S_i, S_j \in S$, if $S_i \neq S_j$ and $S_i \cap S_j \neq \emptyset$, the connected components of the boundary $S_i \cap S_j$ are either single points or topologically equivalent to a segment.*

Proof. By contradiction, assume for some S_i, S_j, there is a component $\mathcal{B} \subset S_i \cap S_j$ that is neither a point or a topological segment. Since $\mathcal{B}$ is a 1-dimensional manifold, then it must be a closed curve, homeomorphic to a circle. Since both S_i and S_j are topological disks (whose boundary is homeomorphic to a circle), then $\mathcal{B} = S_i \cap S_j$. Consequently, $S_i \cup S_j$ is homeomorphic to a sphere without any boundaries, meaning that $S_i \cup S_j = \mathcal{M}$. Thus, $\{S_i, S_j\}$ is a segmentation of $\mathcal{M}$, which contradicts $|S| \geq 3$. □

Theorem 2. *Let S be a segmentation of a connected surface $\mathcal{M}$ into topological disks, such that $|S| \geq 3$, and let $\mathcal{G}$ be the dual multi-graph induced by S. Let us consider a spanning tree $\mathcal{T}$ for the multi-graph $\mathcal{G}$, and consider the set $\mathcal{F} = \mathcal{E} \setminus \mathcal{T}$ of edges that are not in $\mathcal{T}$. The boundary components associated to the edges in $\mathcal{F}$ form a curves network $\mathcal{B}_\mathcal{F}$ that cuts $\mathcal{M}$ into a topological disk.*

Proof. We first consider that each edge $e \in \mathcal{E}$ corresponds to a boundary component $\mathcal{B}_e$ homeomorphic to a segment. Indeed, by Lemma 1, $\mathcal{B}_e$ can only be a topological segment or a single point, but the latter is excluded by construction.

Let us now construct $\mathcal{M}'$ by merging regions in S along the boundary components associated to edges in $\mathcal{T}$. We start with $\mathcal{M}_0 = S_0$, and inductively show that $\mathcal{M}_i$ is homeomorphic to a disk.

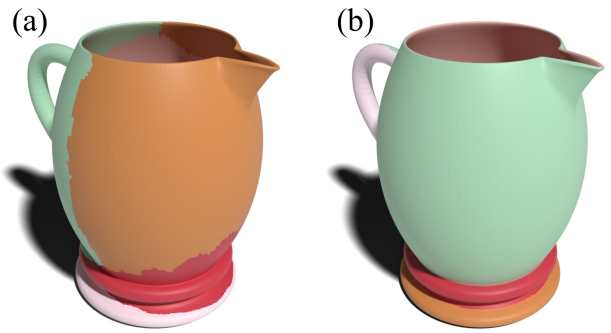

Figure 8: Left (a): Voronoi decomposition of a surface using geodesic edge weights, as in Equation (5). Right (b): the same surface and number of points, but using the angular weights, as in Equation (6).

$\mathcal{M}_0$: Since $S_0 \in S$, then $\mathcal{M}_0 = S_0$ is a topological disk.

$\mathcal{M}_i$: Let $e \in \mathcal{T}$ be an unvisited edge whose associated boundary component $\mathcal{B}_e$ belongs to a region in $\mathcal{M}_{i-1}$, and let us consider the unvisited region S_e incident on the boundary component $\mathcal{B}_e$. Surface $\mathcal{M}_i$ is obtained by gluing $\mathcal{M}_{i-1}$ to $\mathrm{S_e}$ along the boundary component $\mathcal{B}_e$. By the inclusion–exclusion principle for the Euler characteristic, it follows

$$\chi(\mathcal{M}_i) = \chi(\mathcal{M}_{i-1}) + \chi(S_e) - \chi(\mathcal{B}_e) = 1\,, \quad (4)$$

meaning $\mathcal{M}_i$ is topologically equivalent to a disk.

We remark that $\mathcal{M}$ is connected, meaning that any two regions S_i and S_j can be joined by a path crossing a finite number of codimension-one boundaries, hence $\mathcal{G}$ is connected as well. Consequently, since $\mathcal{T}$ is a spanning tree, it connects all the regions in S.

We now prove that $\mathcal{M}' \cup \mathcal{B}_\mathcal{F} = \mathcal{M}$. Let $p \in \mathcal{M}$ be any point on the surface. If p belongs to the interior of some region S_i, it also belongs to $\mathcal{M}'$ and we are done. Otherwise, let S_i and S_j be two regions such that $p \in S_i \cap S_j$. If p belongs to a component of the boundary that is homeomorphic to a segment, it either belongs to $\mathcal{M}'$ or to $\mathcal{B}_\mathcal{F}$ and we are done. If that is not the case, there must be a third region S_k such that a connected component of the boundary $S_i \cap S_k$ has p as an extrema, which reduces to the previous case.

Since $\mathcal{M}'$ is homeomorphic to a disk and $\mathcal{M}' \cup \mathcal{B}_\mathcal{F} = \mathcal{M}$, then $\mathcal{B}_\mathcal{F}$ cuts $\mathcal{M}$ into a topological disk. □

B. Metrics

As discussed through the paper, our pipeline can be tuned with arbitrary metrics to identify distances, merging scores, and boundary weights. Here, we briefly discuss the different alternatives we propose in our implementation, but we stress that more sophisticated application-dependent metrics can be easily applied as well.

B.1. Distance function

The Voronoi decomposition of a surface is defined for any valid distance function that satisfies the metric axioms [20, §1]. However, to guarantee the scalability of our algorithm, we rely on the implementation of Voronoi decomposition provided in [38], which binds us to the application of Dijkstra's algorithm on the dual mesh-graph. In order to evaluate our algorithm under different settings, we consider two alternative weight functions for the edges. A comparison between the two weight functions is depicted in Figure 8.

Geodesic distance. While the standard L_2-norm is sufficient to represent geodesic distance in the neighborhood of a primal vertex, the dual mesh requires some additional attention. Indeed, a straight edge connecting barycenters of adjacent faces can significantly underestimate their actual geodesic distance. Instead, we use the triangle unfolding method to compute the exact geodesic distance between the barycenters, as described in [58]. For a better numerical stability, we consider the following. Given two flattened adjacent triangles t_{ijk} and $t_{ji\ell}$, the distance between their barycenters $b_{ijk}, b_{ji\ell}$ can be computed as

$$\begin{aligned} \|b_{ijk} - b_{ji\ell}\|_2 &= \left\| \frac{1}{3}(v_i + v_j + v_k) - \frac{1}{3}(v_j + v_i + v_\ell) \right\|_2 = \\ &= \frac{1}{3}\|v_k - v_\ell\|_2 \,. \end{aligned} \tag{5}$$

This allows us to compute the distance between barycenters by handling larger and more numerically stable angles.

Isophotic distance. While the geodesic distance is useful to determine regions by means of their size, many applications require that surface cuts align to sharp features (*i.e.*, areas with high curvature). To this end, we consider using the *purely isophotic distance* [53], a metric based on the variation of the surface normals. Despite being well-defined on continuous surfaces, the no existing solutions address the exact computation of the isophotic distance on triangular meshes.

Instead, we provide an approximation, defined on the dual mesh graph, that exploits the local definition of the metric. We first consider that the surface normal is constant inside a triangle, and thus the isophotic distance between points in the same triangle is zero. When the edge between two triangles is crossed, the dihedral angle is added to the total length of the path.

Since triangular meshes are naturally non-smooth, instead of computing the angle θ between adjacent faces, we directly use the dot product (*i.e.*, $\cos\theta$) to introduce an easing factor. Namely, the weight of the edge e connecting two adjacent triangles t_{ijk} and $t_{ji\ell}$, respectively equipped with normals n_{ijk} and $n_{ji\ell}$, is computed as

$$w(e) = w(t_{ijk}, t_{ji\ell}) = 1 - \langle n_{ijk}, n_{ji\ell} \rangle = 1 - \cos(\theta(e)) \,, \tag{6}$$

where $\theta(e)$ is the dihedral angle between faces incident on e. The weight $w(e)$ assumes value 0 for co-planar triangles and assumes its maximum value 2 when the dihedral angle is π.

The behavior of the isophotic distance can be thus approximated with the Dijkstra distance on the dual mesh graph, where each edge is weighted as in Equation (6).

B.2. Boundary weight

In the discrete setting, a boundary component between two adjacent regions (excluding the degenerate case of regions touching on a single vertex) is a collection of edges. In our implementation, we consider two scenarios for computing cuts on a surface:

- minimizing the length of the cut;
- localizing the cut on sharp features.

Since our cut is obtained from the complementary edges of the minimum spanning tree of the dual multi-graph, we can translate these objectives into easily computable boundary weights.

Minimal length. Minimizing the length of the cut means cutting along the shortest boundary components. Consequently, the spanning tree must contain boundary components that are as long as possible. If we weight a boundary component $\mathcal{B}$ by its length $L_\mathcal{B}$ in 3D, the constraint translates to computing the maximum spanning tree. Equivalently, we can compute the minimum spanning tree if we use as weight the function $w(\mathcal{B}) = L^* - L_\mathcal{B}$, where $L^* = \max_\mathcal{B} L_\mathcal{B}$.

Sharp features localization. On the other hand, localizing the cut on the sharper features is equivalent to keep in the spanning three the less sharp boundary components. To identify the sharpness of a boundary component, we consider the fact that a local flat area around a vertex v is characterized by a null Gaussian $k_g(v)$ and mean curvature $H(v)$. Consequently, said $V_\mathcal{B}$ the set of vertices in the boundary component $\mathcal{B}$, we can describe how flat the surface is around the boundary component using the function

$$\sum_{v \in V_\mathcal{B}} |k_g(v)| + |H(v)| \,, \tag{7}$$

which assumes its minimum value zero if the boundary is near-flat and becomes larger as the curvature values increase. The minimum spanning tree connects the whole surface with an as-flat-as-possible fashion.

B.3. Score function

As mentioned in Section 4, our algorithm for merging region follows a greedy approach that merges adjacent regions if they form a topological disk, preferring pairs that have an higher score. Defining the score is application-dependent, meaning the pipeline can be easily adapted to a specific task. In our experiments, we consider two different approaches to compute the score when merging two regions, depending on the type of distance function used for the Voronoi decomposition.

Minimize perimeter-to-area ratio. When using the geodesic distance, we aim to avoid regions with an elongated shape (*e.g.*, long triangle strips) or presenting choke points (*e.g.*, two large regions connected by a single triangle). To mitigate similar situations and favor the formation of regular regions,

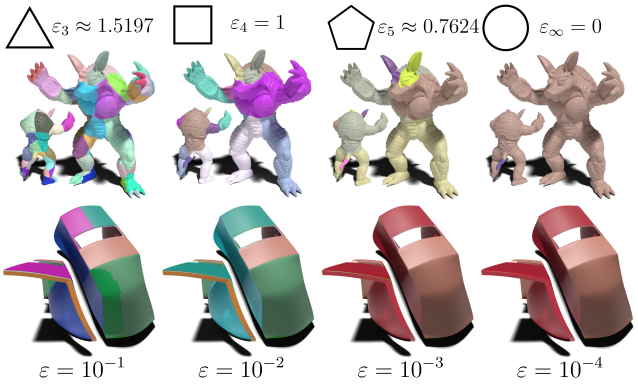

Figure 9: Top row: different segmentations produced by our method at different values of the perimeter-to-area score threshold ε corresponding to different polygons. Bottom row: different segmentations produced by our method for the minimal sharpness score threshold $\varepsilon = 10^{-k}$ at varying of k.

we introduce a score 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

$$\vartheta_{ij} = \frac{\ell_{ij}}{\sqrt{\max\left(A_i, A_j\right)}}. \tag{8}$$

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.

Selecting the threshold to guide the early-stop of the algorithm can be crucial in determining the final result. To understand the intuitive meaning of the threshold ε, we consider that for regular polygons with N sides, Equation (8) evaluates to

$$\varepsilon_N = \frac{2}{\sqrt{N \cot\left(\frac{\pi}{N}\right)}}. \tag{9}$$

The top row of Figure 9 depicts an example decomposition with different threshold values. For our experiments, we consider the pentagon as the reference shape, and thus we consistently use $\varepsilon = \varepsilon_5$.

Minimize sharpness. When the isophotic distance is used, the partitioning tends to align region boundaries along sharper areas of the surface. Here, our goal is to produce regions that are already as flat as possible to simplify the parametrization process. In this sense, we can reuse Equation (7) to characterize the sharpness of the boundary. This requires either changing Algorithm 2 to prioritize the selection of lower scores, or keeping the same interface and using the reciprocal of Equation (7). We decide to use the second approach: said V_{ij} the set of vertices in $M_i \cap M_j$, the score for merging M_i and M_j is given by

$$\vartheta_{ij} = \sum_{v \in V_{ij}} |k_g(v)| + |H(v)|. \tag{10}$$

Differently from the threshold for the perimeter-to-area ratio, defining semantically intuitive reference values for Equation (10) is challenging, but in our experiment we found that values in the family 10^{-k} provide significant differences.

Different example decompositions are depicted in the bottom row of Figure 9. For our experiments, we use the threshold $\varepsilon = 10^{-2}$.

C. Complexity analysis

The fast Voronoi decomposition algorithm introduced in [38], assuming a uniform distribution of the samples and the use of efficient data structures such as hash maps and Fibonacci heaps, has complexity $\mathcal{O}(|V| \log |V| \log k)$, where k is the number of samples and V is the set of vertices of the (dual) mesh. From this, we can derive the complexity of Algorithm 1. Indeed, the algorithm recursively applies the Voronoi decomposition to each region, eventually reaching the worst case scenario where the (dual) mesh is decomposed so finely that each region corresponds to a single (dual) vertex. Thus, to the initial $\mathcal{O}(|V| \log |V| \log k)$ complexity we must add k decompositions of regions that have size $|V|/k$ (assuming uniformity in the sampling). We reach the following recurrence equation.

$$T_{\text{refine}}(|V|, k) = |V| \log |V| \log k + k\, T_{\text{refine}}\left(\frac{|V|}{k}, k\right). \tag{11}$$

By expanding the recurrence we obtain a worst case complexity analysis of Algorithm 1.

$$\begin{aligned} T_{\text{refine}}(|V|, k) &= \sum_{i=0}^{\log_k |V|} |V| \log \frac{|V|}{k^i} \log k \\ &\leq \sum_{i=0}^{\log_k |V|} |V| \log |V| \log k \\ &= \mathcal{O}\left(|V| \log^2 |V| \log k\right). \end{aligned} \tag{12}$$

Once the set S of regions with disk topology is computed, we can either apply the region merging strategy or identify a cut.

For the region merging strategy described in Algorithm 2, we iterate a merging procedure until convergence. In the worst case, this means we apply the procedure until all regions are merged into a single region, which happens after $|S|$ iterations. Since S is a partitioning of the surface into topological disks, it is a proper tessellation of the surface. Consequently, the dual graph has $\mathcal{O}(|S|)$ edges, which is also the number of iterations of the inner loop. Each iteration requires computing the intersection of two region, which can

be done in time $\mathcal{O}(|V|)$ using hash sets, computing a score function, which can be designed to as efficient as $\mathcal{O}(1)$, and pushing values in a priority queue, which can be done in time $\mathcal{O}(1)$ using structures like Fibonacci heaps. Considering that the region merging step can be performed in $\mathcal{O}(|V|)$, as well as the computation of the dual graph, the total complexity of the procedure becomes

$$T_{\text{merge}}(S, |V|) = \mathcal{O}\left(|S|^2\,|V|\right) . \tag{13}$$

Moving to the cut identification algorithm, we consider that the dual multi-graph construction reported in Algorithm 3 is executed with $\mathcal{O}\left(|S|^2\right)$ iterations, but a more sophisticated approach allows its construction through a single visit of the mesh graph in time $\mathcal{O}(|V|\log|V|)$. Always considering that the graph has $\mathcal{O}(|S|)$ edges, the minimum spanning tree can be easily computed in time $\mathcal{O}(|S|\log|S|)$. Recalling that the cut can be executed in time $\mathcal{O}(|V|)$ and that $|S| \leq |V|$ (in practice, $|S| \ll |V|$), we have

$$\begin{aligned} T_{\text{cut}}(|S|, |V|) &= \mathcal{O}(|S|\log|S| + |V|\log|V|) = \\ &= \mathcal{O}(|V|\log|V|) . \end{aligned} \tag{14}$$

D. Practical considerations

When considering the actual implementation, some additional considerations can be made regarding the algorithmic procedures for segmenting and cutting the surface. In particular, we notice that with simple modifications the method can be made more general and efficient.

D.1. Non-manifoldness

Our approach inherently assumes the manifoldness of the surface. This is a strict requirement both for the fast Voronoi decomposition [38] and the disk topology check with the Euler characteristic. However, the nature of the UV parametrization allows us to easily circumvent these issues. Indeed, the mesh can be preprocessed to disconnect multiple triangle fans at the same vertex, and chains of non-manifold edges can be duplicated to disconnect the triangles incident on them and forming new boundaries [3]. All these operations eventually produce a manifold surface $\mathcal{M}'$ with the same triangles as the original non-manifold mesh $\mathcal{M}$. Consequently, the parametrization we obtain for the $\mathcal{M}'$ is also valid for mesh $\mathcal{M}$.

D.2. Initial samples

While the subsampling step can be made rather efficient and guarantees a disk decomposition of the surface, the checks for the Euler characteristic of each region still introduce some computational overhead with respect to the initial sampling. Especially when dealing with complex high-genus surfaces, having a reasonable initial guess can significantly ease the computation. However, to the best of our knowledge, the minimum number of topological disks that cover a surface of arbitrary genus g is not known. Thus, we consider a simple strategy to estimate an initial number of samples $N(g)$, given the genus g of the surface. From Figure 10a, it is easy to verify that a torus can be easily decomposed into four regions with disk topology, meaning $N(1) = 4$. Given a surface of genus g, we can produce a surface $\mathcal{M}$ of genus $g + 1$ via connected sum with a torus $\mathbb{T}$ (see Figure 10b). Since we are adding a torus, we are considering four more regions, but the connected sum removes a topological disk both from $\mathcal{M}$ and $\mathbb{T}$, resulting in a net increase of two regions ($N(g+1) = N(g) + 2$). By solving the recurrence, we get $N(g) = 2(g+1)$, which is a good fit also for the sphere (two disks are sufficient to cover a sphere).

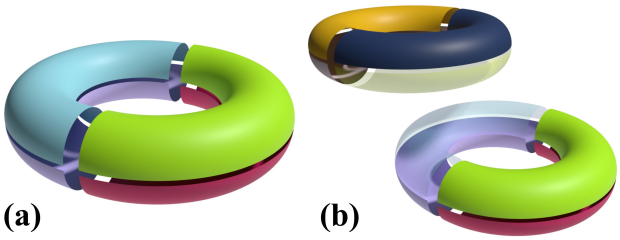

Figure 10: Left: a torus decomposed into 4 topological disks. Right: the connected sum of two tori produces a surface of genus 2, resulting in the addition of three regions and the removal of one (a net increase of two regions).

Filippo Maggioli is currently a Tenured Assistant Professor at Pegaso University, Department of Information Science and Technology. Previously, he was Adjunct Professor at Pegaso University, Postdoctoral Researcher at the University of Milano-Bicocca, and Research Intern at the King Abdullah University of Science and Technology. Filippo obtained is M.Sc. degree in Computer Science in 2019 at Sapienza – University of Rome, where he also defended his Ph.D. thesis on "Scalable Geometry Processing for Computer Graphics Applications" in 2023. His research activity is in the fields of Computer Graphics and Computer Vision, with a particular focus on Geometry Processing, Computational Geometry, Shape Analysis, and Spectral Geometry. Other research interests also span the fields of Physical Simulations and Parallel Computing.

Simone Melzi is an Associate Professor at the University of Milano-Bicocca, in the Department of Informatics, Systems and Communication (DISCo), where he leads the 3DiG Lab working on geometry processing, shape analysis and geometric deep learning. Simone was a Post Doctoral researcher in Computer Science at the Sapienza University of Rome in the GLADIA group led by Emanuele Rodolà. Previously, a post-doctoral researcher at the École Polytechnique in the team of Prof. Maks Ovsjanikov (6 months) and at the Università Degli Studi di Verona (2018-2019). He received my PhD in Computer Science at Università Degli Studi di Verona (2018) and graduated in math at the University of Milan "La Statale" (2013). Simone received The Marie-Curie Individual Fellowships and the Seal of Excellence for the H2020-MSCA-IF-EF-ST-2019 proposal NON-LINFMAPS, (score 92.20/100), the BE-FOR-ERC 2020 grant in the role of Principal Investigator as post Doctoral researcher at Sapienza University of Rome, the EG-Italy PhD thesis award (2018) and the Eurographics Young Researcher Award 2023. He is a member of the Junior Fellow of Eurographics and a Scholar of the European Lab for Learning and Intelligent Systems (ELLIS). He works on geometry processing, 3D shape analysis and artificial intelligence. I maintain fruitful collaborations with many world leaders in this area both with academic institutions and companies.