Contents

Seamless Parametrization with Cone and Partial Loop Control

ZOHAR LEVI, Victoria University of Wellington, New Zealand

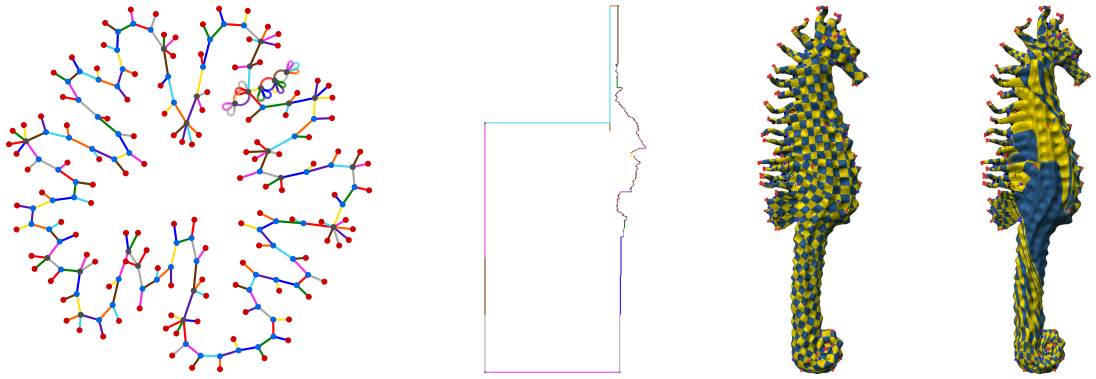


Fig. 1. From left to right: an (abstract) seam graph of the cones over the surface, the domain polygon which the seam is cut into, the final mapping, and [Myles, Pietroni, et al. 2014]'s result. The initial mapping of [Myles, Pietroni, et al. 2014] contains nearly collapsed triangles. These cause numerical issues that the state-of-the-art optimizer [Shtengel et al. 2017] could not handle, and it ended prematurely. Similar to [Levi 2022], our method creates a small metapolygon, consisting only of cone copies, that is optimized before mapping the interior, which alleviates the issue.

We present a method for constructing seamless parametrization for surfaces of any genus, which can handle any feasible cone configuration with any type of cones. The mapping is guaranteed to be locally injective, which is due to careful construction of a simple domain boundary polygon. The polygon's complexity depends on the cones in the field, and it is independent of mesh geometry. The result is a small polygon that can be optimized prior to the interior mapping, which contributes to the robustness of the pipeline.

For a surface of genus > 0, non-contractible loops play an important role, and their holonomies significantly affect mapping quality. We enable holonomy prescription, where local injectivity is guaranteed. Our prescription, however, is limited and cannot handle all feasible holonomies due to monotonicity constraints that keep our polygon simple. Yet, this work is an important step towards fully solving the holonomy prescription problem.

1 INTRODUCTION

In the past decade, advancements in optimization have contributed to the robustness of mapping methods. Methods such as [Lipman 2012; Rabinovich et al. 2017; Shtengel et al. 2017] guarantee foldover-free mappings with low distortion. A prerequisite for these methods is a feasible initial mapping. In unconstrained parametrization, Tutte embedding is a popular choice. However, seamless parametrization, the common approach to quad meshing and construction of seamless atlases, requires that the initial mapping satisfies seamlessness constraints, which calls for a different approach to the problem. In this work, we extend [Levi 2022] that is limited to spheres to any genus.

Given a surface triangle mesh, a parametrization (mapping) assigns UV coordinates to mesh vertices, creating a (flat) *layout* in a planar domain. Before flattening the surface, it is cut into a disk along a tree of seam edges that spans the cones. Each seam edge on the surface is mapped into two *twin (half-)edges* in the domain, corresponding to the two triangles sharing the surface edge. This creates *vertex copies*. Each seam vertex has as many domain copies as its (uncut) seam graph degree. We will refer to the boundary of the domain (disk topology) as the boundary polygon, or polygon for short. We require from a valid mapping to be locally injective, which requires from the polygon to be self-overlapping. A self-overlapping polygon is a polygon that can be triangulated and oriented consistently, and it can admit a locally injective mapping from any disk domain [Weber and Zorin 2014].

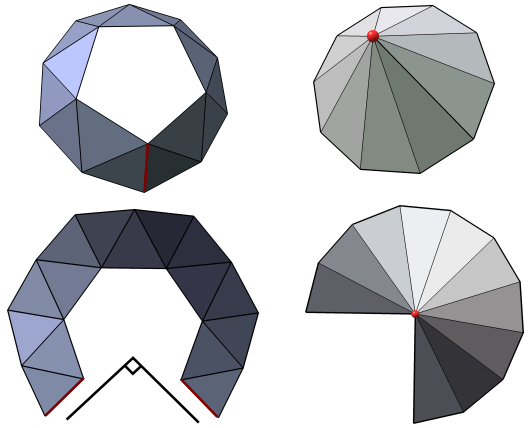


Fig. 2. (Left) a dual loop is flattened isometrically to the plane with a 90° holonomy. (Right) a 3-cone (has a 270° cone angle; see definition 3.1) is flattened with a similar 1-ring holonomy.

We will use the term *holonomy* to characterize a seamless mapping. Consider a closed chain of triangles, where two consecutive triangles in the chain share an edge; see fig. 2. Define a *dual loop* consisting of dual edges that are orthogonal to (and cross) the shared edges. The discrete geodesic curvature at a dual vertex along the loop is the angle defect at the vertex (or the signed angle between the normals of the coinciding dual edges). Alternatively, it is equal

Author's address: Zohar Levi, Victoria University of Wellington, New Zealand.

to the angle between corresponding primal edges. The discrete holonomy angle measures how much the sum of triangle angles along the dual loop (total geodesic curvature) deviates from 360° (angle defect) [Myles and Zorin 2013, 2012]; see fig. 2.

In seamless parametrization, the holonomy of a dual loop in the parametric domain is 90°-multiple. This induces the so called seamlessness constraints that require from the vectors (the difference between end vertices) of two mapped twin edges (in the plane) to differ by a 90°-multiple rotation [Bommes, Zimmer, et al. 2009; Levi 2021, 2022]. As a consequence, the sum of copy angles of a vertex is 90°-multiple. Consider the dual loop that is composed of triangles incident to a vertex (1-ring). The holonomy of the dual loop is equal to the discrete Gaussian curvature of the vertex. A cone singularity is a vertex with non-zero Gaussian curvature in the parametric domain (i.e. not flat—the cone domain angle is different than 360°); see fig. 2.

A common approach to quad meshing is via seamless parametrization, using the following steps ([Bommes, Lévy, et al. 2013; Bommes, Zimmer, et al. 2009]):

- A smooth cross field is generated over the mesh. In addition to a cross per face, a field also defines a matching per edge (or a period jump) that uniquely determines all dual loop holonomies (and cones).
- Seamless parametrization is generated from the field. The surface is cut into a disk, where the seam passes through the cones, and the crosses are used as guiding target frames when optimizing the mapping.
- Cone positions and translations of twin edges are rounded [Campen, Bommes, et al. 2015].
- A quad mesh (similar to a checkerboard texture pattern) with similar holonomies is extracted.

We will use the term *field* to refer only to field characteristics, namely non-contractible loop holonomies and cones (and we ignore the field directions).

Due to the seamlessness requirement, generating an initial seamless mapping for optimization methods is challenging. Recent efforts have been made to generate an initial valid seamless mapping for any (feasible) cone configuration [Campen, Shen, et al. 2019; Levi 2022; Zhou et al. 2020]. Combinatorial approaches, which enjoy robustness in the face of numerical issues, usually construct an auxiliary quad mesh, which a self-overlapping polygon can be extracted from (see [Levi 2021, section 7 in the supplement] for how to lay out a quad mesh in the plane to define a locally injective mapping with a self-overlapping boundary). Then, the input mesh can be mapped into this polygon to generate a valid seamless mapping.

Levi [2022] showed that generating an arbitrary self-overlapping polygon may not suffice since mapping the surface interior is prone to nearly collapsed faces if care is not taken. Instead, the paper takes the approach of constructing a metapolygon (which consists of cone copies) directly. A compact metapolygon allows for optimization prior to mapping the interior [Levi 2022, appendix D], which alleviates the issue. This was the first work to address the full pipeline and demonstrate robustness through extensive experiments. The method also handles any type of cones. Unfortunately, it is limited to spheres.

In this paper, we extend [Levi 2022] to higher genera. We keep most of the pipeline intact and focus on generating the initial metapolygon. Our method enjoys the same benefits that [Levi 2022] offers. These include robustness arising from a combinatorial construction, which is mesh-independent. Similar to [Levi 2022], we offer a numerically robust method to realize the offered combinatorial construction. While our main focus is to generate a locally injective mapping for any given (feasible) set of cones, we take a step further and also prescribe non-contractible loop holonomies.

In a surface of genus ≥ 1, the seam graph that cuts the surface into a disk passes though all cones and a basis of non-contractible loops. A parametrization induces a cone metric, where most of the surface is flat (zero curvature) except for the cone singularities, which can be considered as punctures in the surface. Two homotopic curves on the punctured surface (the mesh without the cones) have the same holonomy in the flat metric. The holonomy (in the flat metric) of a curve is completely determined by the holonomies of a basis of non-contractible loops and a single loop around each cone [Myles and Zorin 2013]. Therefore, to fully characterize a mapping, holonomies of non-contractible loops need to be prescribed in addition to cones. Without prescription, similar to cones, non-optimal holonomies can increase mapping distortion significantly as we show in our experiments.

The main advantage of our approach is in directly constructing a metapolygon, which allows us to control and prescribe loop holonomies. Our prescription, however, is limited by other constraints that we use to generate the polygon, such as monotonicity that serves in keeping the polygon simple. Consequently, we cannot always assign optimal holonomies for all loops, which opens an avenue for future work.

To summarize, given a triangle mesh with a specific set of cone singularities and possibly holonomies of non-contractible loops, we construct a seamless parametrization that is locally injective (guaranteed), set the Gaussian curvature of the cone singularities similar to the input (guaranteed), and set the holonomies of non-contractible loops as close as it can to the input.

1.1 Outline

Our pipeline is similar to [Levi 2022, section 1.2], see summary in section 3.1. The only difference is the metapolygon (definition 3.4) construction that can handle a surface of any genus, which is done in two parts. In the first part, we trace a special seam over the surface. Fields with negative cones (only) are addressed first in section 4. The seam construction includes forming special cone sets and assigning internal and external connections, which specify the seam graph. Fields with positive cones are handled in section 5. In the second part, we solve for the domain metapolygon in section 6, which involves a carefully designed mixed-integer linear program (MILP). Sets of linear constraints are added to the MILP, each responsible for a different aspect of the problem.

The component of the pipeline that generates the polygon, which this paper focuses on, is given in fig. 4. See fig. 3 for a simple example and fig. 10 for a more detailed one. Once the polygon is constructed, the pipeline proceeds as in [Levi 2022, section 1.2].

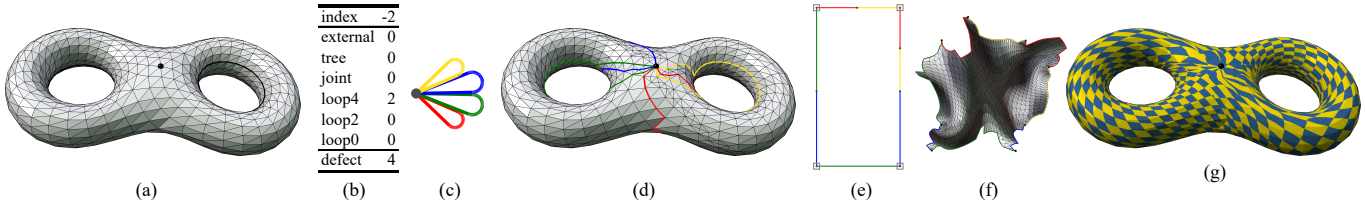


Fig. 3. A simple example of a 2-torus with a single cone. The input is a 12-cone and two tunnel loops (a). A cset is created for the cone. Using alg. 1, an iset is created for the cset. Using alg. 2, an iset chain is created (with a single member), assigning four corners to the iset. According to its index, the iset is even. It is assigned two double loops and four corners. The problem in eqs. (4) to (10) is solved to determine the internal connections of the iset (since the iset consists of a single cone, the solution is trivial). The table in (b) shows the results. The table shows for the iset the assigned connections of a single cone of index -2: two double-loop connections. The defect of the iset is 4, which accounts for four corners and turning number 1 of a self-overlapping polygon. The connection graph is shown (c), and connections are traced over the surface following appendix A. (d) shows the traced seam over the surface. The seam is cut, and the problem in section 6 is solved to determine the metapolygon angles and edge lengths (e). Four corners are marked with squares. Meta-edges between corners are monotone. (f) the UV mapping after interior mapping and optimization. (g) a checkerboard pattern is pulled back to the surface. The high distortion is due to selecting the cone arbitrarily.

Proofs of propositions can be found in appendix E, where appendices C to E can be found in the supplement.

Some of the figures contain vector images, which can be zoomed-in and magnified without compromising quality.

All angles in the paper are measured in degrees.

2 RELATED WORK

We review work related to seamless parametrization and locally injective mappings with guarantees.

Among the various approaches to seamless parametrization [Bommes, Lévy, et al. 2013; Fang et al. 2018; Levi 2021; N. Ray et al. 2006], one popular strategy is generating a smooth cross field over the mesh as a first step [Bommes, Campen, et al. 2013; Bommes, Zimmer, et al. 2009; Campen, Bommes, et al. 2015; Chien et al. 2016; Diamanti et al. 2015; Jakob et al. 2015; Kälberer et al. 2007; Levi and Zorin 2014; Myles and Zorin 2013, 2012; Nicolas Ray, Vallet, Alonso, et al. 2009; Nicolas Ray, Vallet, Li, et al. 2008; Tarini et al. 2011; Vaxman et al. 2016]. Working with a field is convenient since one can easily set cone singularities and non-contractible loop holonomies. In a second step, the mesh is cut into a disk, and the cross field is used as local target frames to lay it out in the plane (parametrization). Most of the previous work, however, does not guarantee in the second step fidelity to the field holonomies generated in the first step, or even worse the resulting mapping may not be locally injective.

Tutte planar embedding [Tutte 1963] and its generalizations [Floater 2003; Gortler et al. 2006] guarantee injectivity, but the convexity requirement restricts the method from satisfying seamlessness constraints. Aigerman and Lipman [2015] generalize Tutte embedding to orbifolds that satisfy seamlessness constraints, but the method is limited to a handful of cone configurations of up to four cones in each.

Recently, Campen, Shen, et al. [2019] proposed a method to construct an initial locally injective mapping that obeys seamlessness constraints. The method can handle any cone configuration except for 1-cones. They start with a discrete conformal mapping [Campen and Zorin 2017], which is seamless up to matching lengths of twin edges. This involves non-linear optimization that is prone to numerical issues. They follow with a padding step to equalize the lengths of twin half-edges, which ensures seamlessness. W. Chen et al. [2019] follow a similar route, using discrete conformal mapping to parameterize a mesh. However, ensuring seamlessness is not detailed.

The approach in [Campen and Zorin 2017] for conformal mapping avoids foldovers by flipping quad edges, during the discrete metric evolution [Springborn et al. 2008], of faces that are about to collapse. This is based on [Luo 2004] that does not provide guarantees that the process converges (avoiding entering an infinite loop of edge flips). Sun et al. [2015] resolved the convergence issue by performing a flip whenever two triangles become concyclic, which ensures that the computation terminates.

Recent work on conformal mapping [Campen, Capouellez, et al. 2021; Gillespie et al. 2021] use Ptolemy rather than Euclidean edge flips, which arise from a hyperbolic perspective. A Ptolemy edge flip preserves an hyperbolic metric and allows to retriangulate a hyperbolic polyhedron without changing its geometry. Both papers report improvement in terms of running time and accuracy. Unfortunately, since Ptolemy flips involve exponential terms, the method is limited to finite precision, and both papers report failures. As pointed out in [Levi 2022, appendix G.2], once the geodesic curvature of the boundary is prescribed (for setting the cone angles), the conformal mapping is unique. The scaling factor can vary drastically over the mapping (e.g. [Levi 2022, figure 16]), which causes numerical issues, specifically nearly collapsed triangles that challenge optimization methods [Shtengel et al. 2017] that use the mapping as initialization. Even if exact arithmetic is used to calculate the conformal mapping, the input to an optimization method needs to be in finite representation, which may contain (nearly) collapsed triangles [Shen, Jiang, et al. 2019].

To circumvent numerical issues, Zhou et al. [2020] combinatorially construct quad patches with the desired singularities. The patches are padded to obtain matching borders, and a full layout (parametrization) of a quad mesh is constructed. A self-overlapping polygon is extracted from the quad layout, into which the input surface can be mapped. The paper reports limited results that include only the auxiliary quad mesh and the initial polygon (without mapping the interior and optimizing the mapping). Similarly to

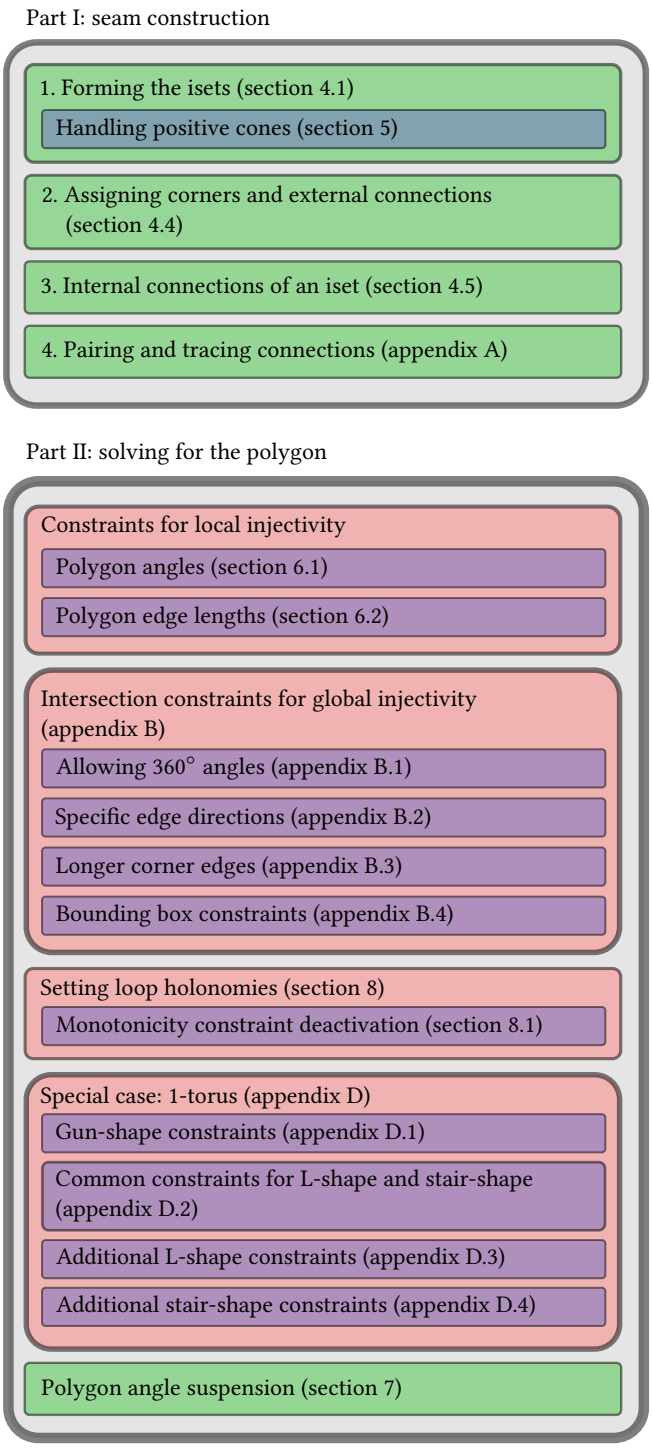


Fig. 4. The pipeline for generating the polygon is divided into two parts. Part I traces a special seam over the surface, and it consists of four steps. Part II solves a MILP, which returns the polygon angles and edge lengths. Each section in this part adds linear constraints to the MILP except for the last one, which suggests an acceleration scheme. Setting loop holonomies is optional, and the 1-torus is treated as a special case with additional set of special constraints.

[Campen, Shen, et al. 2019], the method is limited to cone degree ≥ 2 (definition 3.1).

Since a self-overlapping polygon can be extracted from the layout of a quad mesh, methods that combinatorially construct a quad mesh are relevant to our problem. Grunbaum [1969] generates a quad mesh for a given set of singularities. The paper targets only sphere-topology surfaces and is limited to cone degree ≥ 3. Similarly, Jucovič and Trenkler [1973] generate a quad mesh with the same restriction on the cone degree but for any surface topology. Both papers are purely theoretical: no clear algorithms are given for the suggested constructions, and there are no implementation or experimental details.

Levi [2022] offered the first practical method to treat the full pipeline and showed final optimized mapping results in extensive experiments. The experiments revealed a practical issue that arises in the remainder of the pipeline when using an arbitrary (full) self-overlapping polygon, e.g. one that has been extracted from a quad mesh layout. The issue is that mapping to an arbitrary polygon may result in nearly collapsed triangles, from which a subsequent optimization method cannot recover. The paper overcomes these issues by taking a different approach of directly constructing a compact self-overlapping metapolygon. The metapolygon is optimized using a special routine, which does not scale well to treat large (full) polygons, before proceeding with mapping the interior and the remainder of the pipeline. The method proposed can handle any type of cones (including 1-cones and 2-cones), but it is limited to genus zero.

None of the works mentioned so far can guarantee the prescription of holonomies of non-contractible loops (other than the 90°-multiple constraint that follows naturally from seamlessness). In a concurrent work, Shen, Zhu, et al. [2022] adapt the conformal-based method in [Campen, Shen, et al. 2019] to prescribe (with some limitations) holonomies of non-contractible loops with guarantees.

The work of Myles, Pietroni, et al. [2014] is relevant to our context since it provides guarantees for mapping injectivity. It starts by tracing a motorcycle graph over the surface to create an initial quad patch partition. This is followed by an algorithm to modify the quad layout structure to ensure consistent parametric lengths. Finally, each patch is mapped to create an initial locally injective mapping. While the algorithm guarantees a valid mapping for any given field, it does not guarantee to preserve the input field singularities. Specifically, it may add cones to the final mapping, thus violating our requirement to find a mapping for a given set of cones, which were obtained, for example, by post processing (e.g. to ensure minimal cone distance) or careful placement by an artist. Because the method traces a given field, it is likely to produce a mapping with similar loop holonomies, but there is no guarantee of that. Apart from that, the method generates a mapping that contains nearly collapsed triangles. If nothing else, [Myles, Pietroni, et al. 2014] provides a base for comparison.

Herein, we generalize [Levi 2022] to surfaces of any genus. Moreover, we take advantage of the direct control over the metapolygon our method provides to set non-contractible loop holonomies.

3 BACKGROUND

A cone singularity of a triangle mesh translates into a singular vertex in the generated quad mesh, with corresponding graph degree. Considering the relation between cone singularities in a cross field and quad mesh singularities, we will use the following definition.

Definition 3.1. A k-cone is a cone singularity that has the equivalent following properties:

- A corresponding vertex with graph degree k in the (generated) quad mesh (or *quad degree* for short).
- A (domain/field) cone angle $k \cdot 90°$.
- Field index $1 - \frac{k}{4}$ if it is an inner vertex, and $1/2 - \frac{k}{4}$ if it is a border vertex. The field index determines if a cone is negative or positive (field index, and zero means it is a regular vertex).
- (Domain) angle defect $360 - k \cdot 90°$ if it is an inner vertex and $180 - k \cdot 90°$ if it is a border vertex.

In the figures, the vertices are color coded: 3-cones in red, 5-cones in blue, 6-cones and higher (quad) degree in black, 2-cones and 1-cones in magenta, and special regular vertices in green.

3.1 The Sphere Case

We summarize the proposed algorithm for the sphere case in [Levi 2022]. The input to the algorithm is a mesh (sphere topology) with prescribed (desired) cone singularities. The output is a locally injective seamless parametrization with the corresponding cone angles. The main part of the algorithm is constructing a domain polygon, and we start by recalling some related definitions.

Definition 3.2 (polygon). We will use the term polygon to refer to the domain boundary polygon of a mapped disk (cut mesh).

Definition 3.3 (metagraph). We define cones and possibly a single special regular vertex as metavertices. A meta-edge between two metavertices will refer to a path of regular vertices (that are omitted from the description) between them. Metavertices and meta-edges constitute a metagraph.

Definition 3.4 (metapolygon). A description of a boundary polygon that consists only of copies of metavertices. It is a cut of the surface along a seam metagraph and laid out in the plane. It satisfies seamlessness constraints and respects the (total) cone angles of the corresponding surface vertices. We distinguish it from a *full polygon* that also includes all the copies of regular vertices on the boundary of the cut mesh (the seam).

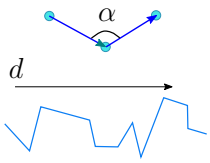
Definition 3.5 (polygon vertex angle). In the context when there is no triangulation (the interior is not mapped yet), the angle α at a (CCW-oriented) polygon vertex is the *internal* angle between the two incident edges. The internal (unsigned) angle α of a polygon (or polyline) vertex is to the left of the incident edge vectors.

Definition 3.6 (monotone polyline). A polyline is monotone with respect to a direction d if the (unsigned) angle between a polyline edge direction and d is at most 90°.

See inset for illustration of a polyline vertex angle (*top*) and a monotone polyline (*bottom*).

Definition 3.7 (field index of a set of cones). The field index, or index for short, of a set of cones is the sum of (field) indices of the cones in the set. The definition applies recursively to a set of sets of cones. We define the index function idx $(\cdot)$ that returns an index of a cone or a cone set.

While [Levi 2022] worked with field index, e.g. in the context of grouping cones into sets of total field index 0 (used to enforce polyline monotonicity), we work more accurately with *defect*.



Definition 3.8 (domain defect). By defect, we will refer to domain defect by default, and its units are 90°. Let v_i be the ith polygon vertex (a cone copy) with (inner) angle α_i. We define the defect function as:

$$\Delta(v_i) \coloneqq 2 - \frac{\alpha_i}{90°} . \tag{1}$$

Let v be a cone (on the surface) with seam graph degree n. Let α be its (prescribed 90°-multiple) angle and $\{v_i\}$ be the set of its n copies. We define its defect as:

$$\Delta(v) \coloneqq \sum_{i=1}^{n} \Delta(v_i) = 2n - \frac{\alpha}{90°} = 2n - 4\left(1 - \mathrm{idx}(v)\right) . \tag{2}$$

The defect of a set of cones is the sum of cone defects, and the definition applies recursively to a set of sets of cones.

Definition 3.9 (balanced set). We will call a set with index zero a balanced set. By balancing a set, we will refer to making its index zero. When we discuss connections in section 4, we will also use the term balancing the defect of a set, which will refer to setting its defect to zero. From context, it will be clear what we refer to by balancing, or else it will be noted explicitly.

In our construction, since the seam is a single connected component, sets with index zero have the required two connections to set their defect to zero (see section 4.3). Hence, it will be convenient to consider a (connected) balanced set as having both index zero and defect zero (as implicitly assumed in [Levi 2022]).

The heart of the algorithm that was proposed in [Levi 2022] is the construction of a self-overlapping polygon, which is necessary for a locally-injective mapping [Weber and Zorin 2014]. Identifying a self-overlapping polygon is not an easy task, much less constructing one. However, a simple polygon is trivially self-overlapping.

The approach that [Levi 2022] takes is to construct a simple polygon. This is done by first constructing a special seam that ensures that the seam degree of each cone is in a proper range: a cone has enough copies to partition its angle between them such that each angle is $\leq 360°$, as required from a simple polygon. Cones are grouped into balanced sets, and their copies are (mostly) in consecutive order in the polygon. This permits angle assignment that ensures monotone polylines. The defect of a (connected) balanced set is zero, and the number of left turns in its polylines equals the number of right turns. If the right and left turns are synchronized properly, then monotonicity can be achieved. Constructing the polygon from monotone polylines prevents self-intersections and ensures its simplicity.

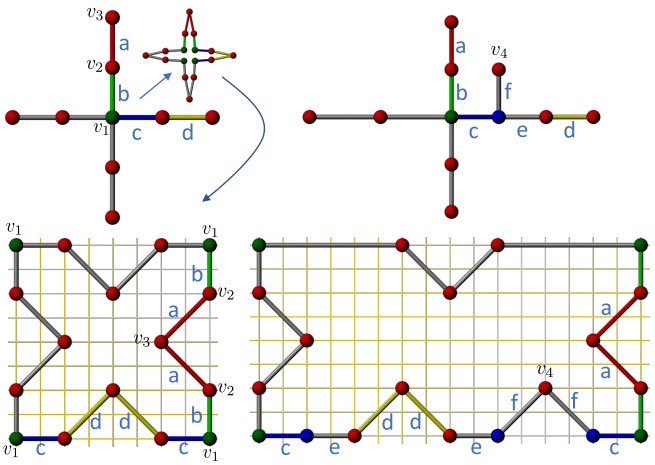
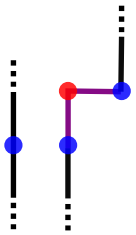


Fig. 5. An example of the sphere case. *Top*: the (uncut) seam graph (over the surface). An auxiliary regular vertex in green, a 3-cone in red, and a 5-cone in blue. *Bottom*: the domain metapolygon after cutting the seam at the *top* image. *Left*: the foundation set that is composed of eight 3-cones and an auxiliary regular vertex (defect 0). *Right*: the foundation set with an added pair of {3,5}-cones that contributes to the polygon two monotone polylines. For more details, see [Levi 2022, figure 11].



A simple example is a balanced set with $\{3, 5\}$-cones. On the surface, the seam goes through the set's cones consecutively. When the seam is cut, the copies of the set appear on two polylines, each containing a series of polygon vertex angles. Since the set is balanced, we can assign angles to the set copies such that the number of left turns is equal to the number of right turns in each polyline, creating two monotone polylines; see inset. Notice that the total defect of each polyline is zero. Moreover, the defect of each sequence of angles is in $[-1, 1]$, which ensures monotonicity.

A simple polygon has turning number 1 or defect 4 (360°). According to the Poincaré–Hopf theorem, there must be a set of cones that cannot be balanced, with index equals to χ (the Euler characteristic), which we term a *foundation set*. Its copies constitute the so called *foundation polygon*, which is a minimal simple polygon that accounts for the turning number.

Definition 3.10 (foundation set). A foundation set on a closed surface of genus g is a (minimal) set of cones with index $\chi = 2 - 2g$, whose copies comprise the foundation polygon. According to the Poincaré–Hopf theorem, the total index of all vertices of a surface is χ, hence there must be such a subset.

Definition 3.11 (foundation polygon). A metapolygon that consists only of the vertex copies of a foundation set.

Monotone polylines of balanced sets are added to the foundation polygon, where monotonicity keeps the polygon simple; see fig. 5 for illustration. In the following, we describe the steps of the pipeline.

Creating the foundation polygon. A *foundation set* of cones is identified, where there are 10 possible sets [Levi 2022, section 5]), and a *foundation polygon* is constructed. The metapolygon is arbitrary, and all 10 options are provided. (In practice, it is not constructed explicitly, and only its angle assignment—given as a simple rule—is used later when constructing the metapolygon for all the cones.)

Forming the sets. After identifying the foundation set, the rest of the cones are divided into balanced sets. The purpose, again, is to add them to the polygon as monotone polylines that will not violate its simplicity.

The seam is traced to connect the sets. The tracing is done according to specific refinement rules that keep the refinement moderate.

Assigning polygon angles and edge lengths. Once the seam is cut, a boundary metapolygon emerges, but we only have the order of its vertices. A linear program is solved for the polygon's angles that ensure polyline monotonicity. A second linear program is solved for valid edge lengths.

Improving the polygon. To increase the quality of the mapping, the metapolygon is improved by maximizing the minimal angle of its triangulation that is used as convex subdomains for mapping the interior (non-boundary vertices) in the next step. If a triangulation contains a sliver (with an angle close to zero), then mapping into such a subdomain (with the sliver as boundary) is bound to result in (nearly) collapsed triangles, which would pose a challenge (due to numerical issues) in the final optimization.

The algorithm is based on nonlinear programming, and it does not scale well. Therefore, it is imperative that it is applied to a (small, coarse) metapolygon rather than a full one.

Mapping into a self-overlapping polygon. [Levi 2022] offers a scheme similar to [Shen, Jiang, et al. 2019] to map the interior of the mesh. Another alternative is dual-harmonic mapping [Weber and Zorin 2014].

Simplifying and optimizing the mapping. The initial valid mapping is optimized to lower distortion using state of the art, e.g. [Shtengel et al. 2017]. While distortion is lowered, most, if not all, of the refinement that was introduced when tracing the seam is removed.

Our algorithm takes a similar approach, using a similar pipeline. The input to the algorithm is a closed mesh of genus ≥ 1 with prescribed (desired) cone singularities and non-contractible loop holonomies. The output is a locally injective seamless parametrization with the corresponding cone angles that are satisfied exactly and holonomies that may be fulfilled only partially. We begin by constructing a simple foundation metapolygon that accounts for χ and the polygon turning number. In the sphere case, there are only 10 possible foundation sets, and an arbitrary construction was offered for them. However, this number grows with the genus. Moreover, for genus > 0, non-contractible loops come into play, and they balance cone index.

Solving for the metapolygon angles and edge lengths is done simultaneously due to the presence of non-contractible loops that may lead to polygon configurations with valid angles that do not have corresponding edge lengths.

Incorporating non-foundation cones is done similar to the sphere case, using balanced cone sets, and poses no further complication except for the special case of genus 1.

Once the metapolygon is constructed, the rest of the pipeline proceeds unchanged. In summary, the focus of this paper is to construct a simple self-overlapping (foundation) polygon for a mesh of genus ≥ 1.

3.2 Additional Definitions

We handle a closed surface of genus g, a g-torus, that has a basis of $2g$ non-contractible loops. We will refer to these simply as loops. In the following sections, we treat the case of $g \geq 2$ and defer the case of $g = 1$ to appendix D.

Definition 3.12 (tunnel and handle loops). A closed surface $\mathcal{M}$ separates $\mathbb{R}^3$ into two parts: interior $\mathbb{I}$ and exterior $\mathbb{O}$ such that $\mathbb{I} \cap \mathbb{O} = \mathcal{M}$ and $\mathbb{I} \cup \mathbb{O} = \mathbb{R}^3$. A loop on $\mathcal{M}$ is a handle loop if it is trivial in $\mathbb{I}$ but non-trivial in $\mathbb{O}$. Similarly, it is a tunnel loop if it is trivial in $\mathbb{O}$ but non-trivial in $\mathbb{I}$. For details and illustration see [Dey et al. 2013, section 2.1].

The input to the algorithm is g (out of $2g$) disjoint loops, e.g. g tunnel loops. These loops can be acquired by different means (we used [Dey et al. 2013]), and they are not related to the problem nor require user intervention. The algorithm traces additional g (e.g. handle) loops that complete a non-contractible loop basis. The result is g *double loops*.

Definition 3.13 (double loop). A double loop is a pair of loops that is composed of a tunnel loop and its associated handle loop, which (inevitably) intersect.

Definition 3.14 (modifying a loop to pass through a vertex). Let ℓ be a loop and v be a vertex. The operation of modifying ℓ to pass through v refers to finding a loop ℓ' that passes through the vertex such that:

- ℓ and ℓ' are in the same homotopy class, $\ell' \in [\ell]$.
- In addition to passing though v, ℓ' passes through the same set of metavertices (cones) that ℓ passes through, and it preserves its intersections in the seam graph (except possibly at v).

Definition 3.15 (unit triangular matrix). A lower triangular matrix filled with ones (on the diagonal and below).

For some of the constraints in the (integer or mixed integer) linear programming problems, we hint that they were formulated using the Big M constraint approach [D.-S. Chen et al. 2010, section 3.6.1].

4 SEAM CONSTRUCTION FOR NEGATIVE FIELDS

We begin by describing the construction of our special seam for an input field that consists of a set of negative cones (only), which we term *negative fields*. In this setting, the *foundation set* consists of all the cones in the field.

For a 1-torus, there is only one cone set, the empty set. For genus $g \geq 2$, the number of negative cone sets grows with the genus. For example, for genus 2–5, the number of possible negative sets in each case is 22, 231, 1,575, and 8,349 respectively (10,177 in total). Unlike the sphere case, which has only 10 cases that could be handled arbitrarily, here a more methodical approach is needed. For low genus, the number of cases is still reasonably low, and we use that later to perform exhaustive testing of the method.

To generalize the approach and reduce the infinite number of cases that need to be handled, we perform two simplifications. The first simplification is to group cones (that may have fractional index) into sets with integral total index.

Definition 4.1 (cset). A cset (short for cone set) is a set of cones. We will use two types of csets:

(i) A negative-dominant cset that consists of a single negative cone and possibly additional positive cones. The index of the cset is negative. The foundation set is partitioned into negative-dominant csets.
(ii) A balanced non-foundation cset.

The concept of a cset is introduced to conveniently incorporate positive cones later (section 5) with minimal modification to an algorithm that expects negative cones only.

Definition 4.2 (iset). An iset (short for integer set) is a set of csets. Its index is integral and equals the sum of the csets' indices. An iset is either *odd* or *even*, depending on its index.

The second simplification is to divide the csets into isets that are classified and treated based on their index modulo 2. An iset's index is adjusted to be modulo 2 by assigning to it double loops, each equivalent to index 2 (section 4.3). Details are given in definition 4.7.

Our objective becomes creating balanced isets (by connecting their cones with the seam), which have adjusted index 1 or 2.

Definition 4.3 (balanced iset). A balanced iset will refer to an iset with index zero (total index of cones and double loops) and defect zero (connected).

The allotted (determined by the genus) double loops balance all isets except for either one or two (section 4.3). These exceptional isets are termed *corner isets* (definition 4.5), and they contain excess defect that accounts for the metapolygon's turning number 1 (total angle defect 360°). All other balanced isets add monotone polylines (each has total angle defect 0°, i.e. like a straight line, there is no change of direction) to the foundation polygon.

The two simplifications above reduce the number of iset cases from infinite to 76 (section 4.5). This small number of cases are handled conveniently with a specialized *integer linear program* (ILP).

This section describes how to trace the seam over the surface, connecting the cones and non-contractible loops into a single connected component, which is summarized next. First, the isets are formed (section 4.1). Then, connections are assigned.

We distinguish between two forms of connections: an external connection that connects between two cones from two different isets (i.e. connects two isets), and an internal connection that connects cones from the same iset. A chain of isets is created, which assigns corners and external connections to the isets (section 4.4). This is followed by assigning internal connections (to cones) in each iset (section 4.5), which ensures the necessary connectivity and defect for each cone (section 4.3). This is done through solving an ILP, which also assigns (iset) external connections to specific cones in the iset.

Initially, connections are only assigned (to isets and cones). After the connection assignment is complete, the specific pairing of connection ends and the tracing itself are performed (appendix A). The only exception to that is an external joint-loop connection (section 4.2) between an odd pair of isets in alg. 2, which is immediately traced over the surface.

The ability to assign only a connection end to a cone and defer the decision of which cone it is connected to and via what path comes from the combinatorial nature of the algorithm. From the algorithm point of view:

- All cones of the same degree are the same.
- All free loops (definition 4.4) are the same.
- All homotopic paths are the same.

Therefore, if we have, for example, an iset with four cones, where each is assigned an internal tree connection (section 4.2), then it does not matter between which cones the two tree connection paths are traced as long as each cone receives a single tree connection. Similarly, when assigning external tree connections in an odd iset chain (section 4.4), the order of the isets in the chain does not matter as long as each iset receives its assigned amount of connections.

Similar to [Levi 2022], the combinatorial nature is exploited to keep the seam short. When part of the seam-construction algorithm requires a cone, an arbitrary free one—that has not been connected to the seam yet—with the requested quad degree is picked. Later, before tracing the first connection path to the cone (e.g. by modifying a loop to go through it or connecting it to another cone), the cone is replaced with a different free cone (if there is one) of the same quad degree that is closest to the source of the path (or loop). A similar approach is taken with the free loops that have no "identity" until connected (definition 4.4).

4.1 Forming the Isets

We partition the foundation set into negative-dominant csets (definition 4.1). This is trivial under the current setting (a negative field): we create a cset for each cone. Next, we divide the csets into isets (definition 4.2), which is described in alg. 1.

Alg. 1 groups cones into isets, according to the fraction part of their index (specifically, their index modulo 0.25). It distinguishes between seven types of isets, and it operates in a greedy way, covering all possible options.

4.2 Connection Types

Before specifying how to connect the cones, we describe in this section the types of (graph) connections that we use. The variety of connection types provides the flexibility of how to divide a given number of connections between cones.

Definition 4.4 (loop ownership). In the algorithm, we assign a double loop to a vertex (or two vertices in the case of a joint loop below), which becomes its *owner*. Connecting an assigned double loop to its owner is done by modifying the loop to pass through the vertex (definition 3.14). A specific double loop that will be connected is chosen arbitrarily from the pool of free loops—loops that are not yet owned by and connected to any metavertex—which is initialized to the g input loops (each representing a double loop). A second loop is traced such that the vertex (or two vertices in the case of a joint loop below) is at the intersection of the two loops. A set that contains a vertex that owns a double loop is said to own the double loop.

Algorithm 1: Forming the isets

Input: a set of foundation csets
Output: a set of isets

```
Partition the csets into four queues Q_0, ..., Q_3, according to their index modulo 0.25
while ∃i, |Q_i| > 0 do
    Create a new iset I
    if |Q_0| > 0 then // 0
        I.add( Q_0.pop() )
    else if |Q_3| > 0 and |Q_1| > 0 then // 0.75 + 0.25
        I.add( Q_3.pop() )
        I.add( Q_1.pop() )
    else if |Q_3| > 1 and |Q_2| > 0 then // 2 × 0.75 + 0.5
        for j ← 1 to 2 do
            I.add( Q_3.pop() )
        I.add( Q_2.pop() )
    else if |Q_3| > 3 then // 4 × 0.75
        for j ← 1 to 4 do
            I.add( Q_3.pop() )
    else if |Q_2| > 0 then
        I.add( Q_2.pop() )
        if |Q_2| > 0 then // 2 × 0.5
            I.add( Q_2.pop() )
        else // 0.5 + 2 × 0.25
            for j ← 1 to 2 do
                I.add( Q_1.pop() )
    else // 4 × 0.25
        for j ← 1 to 4 do
            I.add( Q_1.pop() )
```

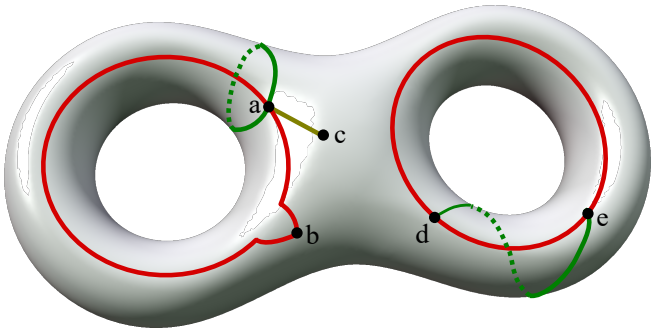


Fig. 6. Connection types. Vertex a owns a double loop. Vertices a and b are connected via a 0-2 connection, where a is the provider, and b is the receiver. Vertices a and c are connected via a tree connection. Vertices d and e are connected via a joint-loop connection.

The seam is initialized as the g input disjoint loops, and g more loops are traced by the algorithm. These are primal closed paths that are marked as seam, e.g. fig. 7(a). The final seam that is traced over the surface connects the cones and passes through a basis of $2g$ non-contractible loops, thereby cutting the surface into a disk, with all cone copies on its boundary. We define four types of connections (meta-edges between cones) and list their contribution to the graph degree of the cones that are involved in them. See fig. 6 for illustration.

A double-loop connection. A vertex that *owns* a double loop serves as the base point—the intersection of the two loops. It can be viewed as a (double) self-connection. A double-loop connection consists of four connections, and the contribution to the degree of the vertex is four. The connection is implemented by modifying one of the g input loops (that has not been assigned yet) to pass through the vertex (definition 3.14). Then, another loop is traced from the vertex to itself, from one side of the loop to other (without intersecting other parts of the seam).

A 0-2 loop connection. Connecting a vertex u that owns a double loop with another vertex v. One of the two loops is modified to pass through v (definition 3.14). u is said to provide a 0-2 loop connection, and v is said to receive one. The same terminology applies to sets containing the vertices. The contribution to the degree of v is two. There is no contribution to the degree of u.

A tree connection. Connecting two vertices via a seam metagraph edge, which contributes one to the degree of each vertex.

A joint-loop connection. Connecting two vertices u and v via a double loop. The two vertices share ownership of the double loop. The contribution is three to the degree of each vertex. When connecting the assigned double loop, a free input loop is selected, e.g. a tunnel loop t. We modify t to pass through u and then through v (definition 3.14). We trace a new handle loop h from u to v that starts on one side of t and ends on the other. When tracing h (as a path over the surface), we avoid intersecting other parts of the seam (except for the path's end points). Since the other $g-1$ disjoint loops are marked by the seam and we avoid intersection with other parts of the seam, tracing a path from one side of the loop to the other results in a loop in the correct homotopy. The two loops overlap, and in a generated abstract graph drawing they look like three edges between u and v (each vertex receives three connections), e.g. see fig. 8 (right), which contains three joint-loop connections.

4.3 Connections and Defect

Our metapolygon has three possible angles: 90°, 180°, and 270°. As a simple cycle, it has defect 4 (angle defect 360°). Four polygon vertices will be picked, designated as corners, and assigned 90°. The four (meta)polylines between these corners will be monotone, and the first and last edges in each of the four polylines will have the same direction. The metapolygon could be visualized as a coarse rectangle if we scale up the meta-edges that are incident to the four corners and scale down the rest of the polygon edges (e.g. fig. 10(e)). We will refer to the four polylines between corners as *rectangle edges*.

Definition 4.5 (corner iset). An iset with defect > 0. The number of corners in an iset (in all of its domain polylines) equals its defect.

Alg. 2 in section 4.4 designates either a single corner iset (containing four corners) or two corner isets (each containing two corners), depending if there is an odd iset (or they are all even). The total defect of corner isets will be 4, and the rest of the isets will have defect 0 (proposition 4.8). This puts the total defect of the polygon at 4, as required from a self-overlapping polygon (turning number 1).

The polygon is simple. Edge monotonicity ensures that there are no (local) intersections (and the additional constraints in appendix B ensure this globally). The defect plays an important role in guaranteeing monotonicity. Specifically, vertex sequences along the four rectangle edges have defect in $[-1, 1]$. A cone's defect depends on its angle and (seam) degree (see definition 3.8). Cone angles are given (as input), and our special seam sets the proper degree of a cone to ensure that its defect is in the correct range.

From eq. (2), to balance the defect of a vertex v (i.e. $\Delta(v) = 0$) with integer index, we need to set its graph degree (number of connections) to

$$n = 2\left(1 - \mathrm{idx}(v)\right) \ . \tag{3}$$

For example, a regular cone ($\mathrm{idx}(v) = 0$) requires two connections to balance its defect. Each connection (which adds one to a vertex degree in the seam graph) *balances* (sets the defect to zero) 180° or field index 0.5. In other words, increasing a cone's angle by 180° and adding a connection to it preserves its defect. For example, owning a double loop, which contributes four connections to the cone that owns it, has the effect of adding two to the cone's index.

4.3.1 Some Examples. We demonstrate the defect concepts with some examples.

Example 1. A 2-torus (two double loops) with a 12-cone. Assigning a double loop to the cone (definition 4.4) sets its defect to $2 \cdot 4 - 4(1+2) = -4$, and it requires two more connections to balance its defect. Assigning the second double loop to the cone gives it two connections more than required for balancing, and sets its defect to 4. Four of the cone's copies are designated as corners; see fig. 3.

Example 2. A 2-torus with two 8-cones. Assigning a double loop to each cone balances its defect, but the seam remains disconnected. Connecting the two cones with a tree connection connects the two seam parts into a single component. The additional tree connection sets the defect of each cone to 2, and two copies of each cone are designated as corners; see fig. 7.

Example 3. A 3-torus (three double loops) with a 20-cone. Assigning the three double loops to the cone sets its defect to $2 \cdot 3 \cdot 4 - 4(1+4) = 4$. Four of the cone's copies are designated as corners; see fig. 9.

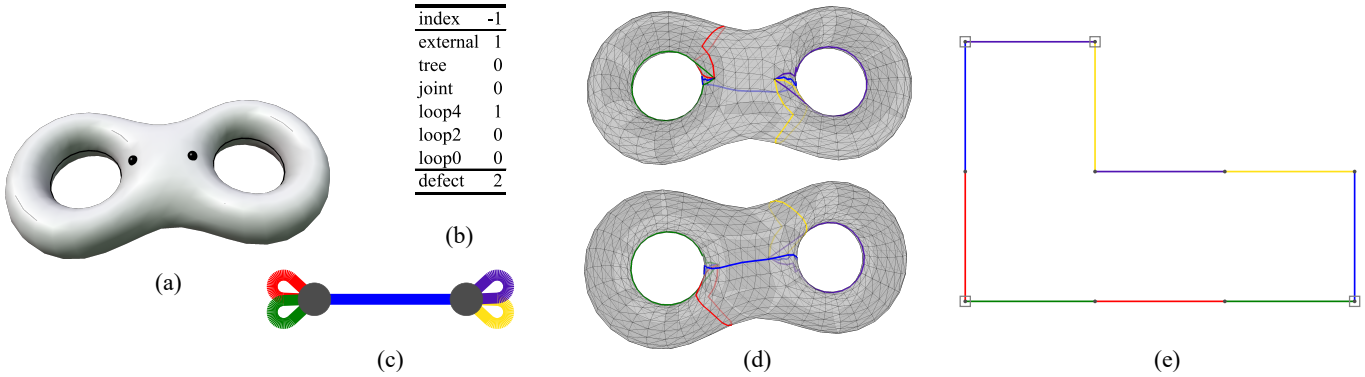


Fig. 7. A simple example of a 2-torus with two cones. The input is two 8-cones and two tunnel loops (a). A cset is created for each cone. Using alg. 1, an iset is created for each cset. Using alg. 2, an iset chain is created, assigning four corners and external connections to the isets. According to their index, the two isets are odd. Each of them is assigned a double loop, an external tree connection, and two corners. The problem in eqs. (4) to (10) is solved to determine the internal connections of each iset (since each iset consists of a single cone, the solution is trivial in this case). The table in (b) shows the results for one of the isets, where both isets share the same characteristics. The table shows for an iset the assigned connections of a single cone of index -1: one double-loop connection and one external tree connection. The defect of each iset is 2, which accounts for two corners, and their total defect accounts for turning number 1 of a self-overlapping polygon. The connection graph is shown (c), and connections are traced over the surface following appendix A, where the isets are externally connected via a tree connection. (d) shows a top and a bottom view of the traced seam over the surface. The seam is cut, and the problem in section 6 is solved to determine the metapolygon angles and edge lengths (e). Four corners are marked with squares. Meta-edges between corners are monotone. Copies of the same cone are in consecutive order along the polygon.

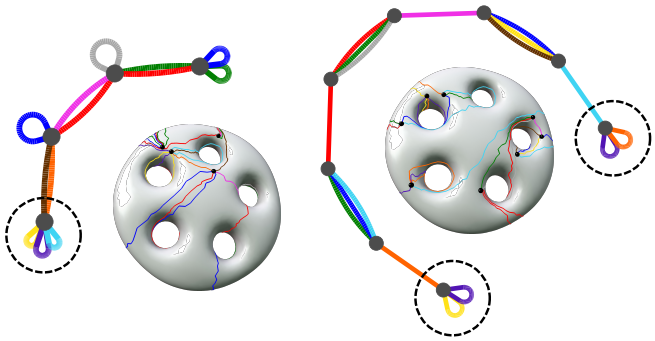


Fig. 8. An odd and an even chain on a 5-torus ($\chi = -8$). Each iset is made up of a single cone. Corner isets are marked with a dashed circle, and all other isets are balanced. (Left) an even chain of four 12-cones. Each iset owns a double loop, and the corner iset owns an extra double loop to account for turning number 1. External connections between isets are 0-2 loop connections. (Right) an odd chain of eight 8-cones. Each corner iset owns a double loop. Otherwise, each odd pair of two cones owns a double loop and is connected via a joint-loop connection. Iset corners and odd pairs are connected via tree connections.

We generalize these examples in the next section by constructing an iset chain.

4.4 Assigning Corners and External Connections to Isets

In the following, we create a chain of isets that consists of isets with defect zero and one or two corner isets. The chain sets the external connections between the isets; see fig. 8. It pairs up *odd isets* and places corner isets at chain ends. This section gives details for the chain construction, and the proof of correctness is deferred to proposition 4.8 after specifying the internal connections (section 4.5) and fully describing the seam.

As an intuition for this step, one can view the isets at this point as containing a single cone each. This step sets the connections between them to achieve the necessary defect. In the next section, the isets are generalized to contain up to four cones each, we solve for their internal connections, and the connections that are set in this section are treated as external.

If there are no odd isets, then the iset chain is (all) even. In this case, we assign four corners to the first iset in the chain (chosen arbitrarily) as well as a double loop to adjust its defect (and it will have defect 4 when fully connected). External connections between isets in the chain will be made via 0-2 loop connections.

If the chain is not (all) even, then there are at least two odd isets. We assign to each of them two corners as well as a double loop to adjust its index (each odd corner iset will have defect 2 when fully connected). Each of the two corner isets is placed at an end of the iset chain (and will be connected to the rest of the chain via an external tree connection). We divide the rest of the odd isets into pairs and connect the two isets in each pair with an external joint-loop connection. For each iset in a pair, we designate its cset with the lowest (negative) *adjusted index* (definition 4.7) as the first cset and use it for the joint-loop connection. This is a greedy choice of assigning the external connection to a cset that needs connections the most to balance its defect. All the odd pairs, corner isets, and even isets are assigned external tree connections.

Definition 4.6 (odd pair). A pair of odd isets that are connected with an external joint-loop connection.

At this point, only external joint connections are made, and the rest of the connections types are set as attributes. The full algorithm

Algorithm 2: Assigning corners and external connections

Input: isets $I_0, \ldots, I_{n-1}$
Output: assigned corners and connections

Sort the isets, prioritizing odd (field) index
$b_{\text{all\_even}} \leftarrow$ false
$i \leftarrow 1$
while $i \leq n$ **do**
  $I \leftarrow I_i$
  if idx $(I) \mod 2 = 0$ **then** // even
    if $i = 1$ **then** // first
      $b_{\text{all\_even}} \leftarrow$ true
      I.loop4conn $\leftarrow 1$
      if $n > 1$ **then**
        I.loop0conn $\leftarrow 1$
      I.corners $\leftarrow 4$
    else
      if b_{all_even} **then**
        if $i < n$ **then** // not last
          I.loop0conn $\leftarrow 1$
        I.loop2conn $\leftarrow 1$
      else
        I.tree1conn $\leftarrow 2$
  else // odd
    if $i \leq 2$ **then** // first two
      I.loop4conn $\leftarrow 1$
      I.tree1conn $\leftarrow 1$
      I.corners $\leftarrow 2$
    else
      $i \leftarrow i + 1$
      $J \leftarrow I_i$
      I.tree1conn $\leftarrow 1$
      J.tree1conn $\leftarrow 1$
      Connect I and J with a joint loop // odd pair
  $i \leftarrow i + 1$

for assigning corners and external connections to the isets is described in alg. 2. In the algorithm, the attributes associated with an iset are the following:

- **tree1conn**: the number of external tree connections to the iset.
- **loop4conn**: the number of double loops that are owned by the iset.
- **loop2conn**: the number of external 0-2 loop connections (section 4.2) that the iset receives.
- **loop0conn**: the number of external 0-2 loop connections that the iset provides.
- **corners**: the number of corners in the iset (definition 4.5).

4.5 Internal Connections of an Iset

Given an odd or even iset I (and its assigned external connections), which consist of n (which is four at most according to alg. 1) csets c_i, we describe how to assign its internal connections. The assignment involves the small integer linear program (ILP) in eqs. (4) to (10). The solution specifies the number of connections of each type that are assigned to each cset. The main objective of the ILP is to maintain metagraph connectivity (a single connected component) and the defect of each cone in the required range for monotonicity (eq. (4) and eq. (2)).

The ILP was designed to specifically address the 76 types of internal iset connections (see proof of proposition 4.8). Therefore, while it may seem that some cases are overlooked, one needs to bear in mind that the problem was reduced to handle only these 76 cases. On the other hand, deep understanding of the constraints is not required for understanding the main algorithm. Therefore, we provide intuition for the constraints, but we do not go into detail (e.g. providing specific cases that need to be filtered out), which would explain the necessity of some of the constraints.

Solving the problem is a convenient way to generate the map between each of the 76 types and its corresponding internal connection configuration (a vector holding the number of assigned connections to each cset—the problem solution), which otherwise can be created manually or acquired from an external source and used as a constant in an application.

Definition 4.7 (adjusted index). The adjusted index of a cset $c \in I$ is:

$$\overline{\text{idx}}\,(c) \coloneqq \text{idx}\,(c) + 2\eta\,(c)\ ,$$

where

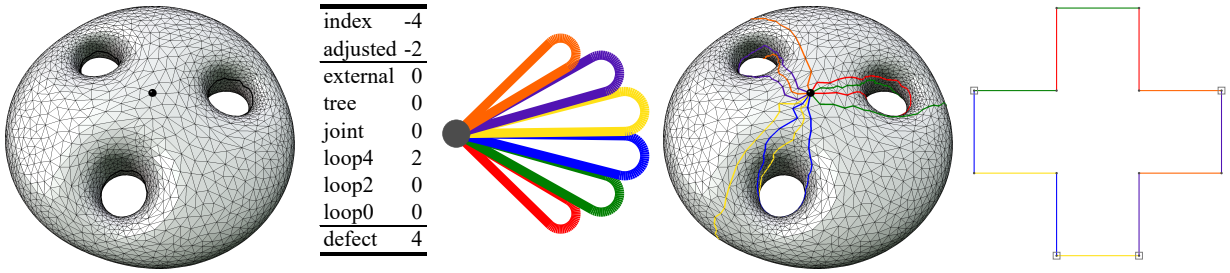
$$\eta\,(c) \coloneqq \left\lfloor \frac{\text{idx}\,(c) + 0.1}{-2} \right\rfloor$$

is the number of *reduced double loops*. These are loops that are connected separately (as double-loop connections) to the negative cone in the cset (which now owns them) when tracing the connections (appendix A). Since $\text{idx}\,(c) \in \{-0.25k \mid k \in \mathbb{N}\}$, we have $\overline{\text{idx}}\,(c) \in \{-0.25k \mid k = 1, \ldots, 8\}$.

If I has an external joint-loop connection (which consists of three connections), then we need to account for it when solving the problem. Instead of introducing a new variable for that, we do the following. We first assign an additional double-loop connection to the iset (++I.loop4conn). This additional double-loop connection (which consists of four connections) is assigned in the problem below to the first cset (which was used for the external joint connection of an odd-iset pair in section 4.4). Then, to compensate for the connection difference between a joint-loop connection and a double-loop connection ($4 - 3 = 1$ connection), we lower the cset's adjusted index by 0.5:

$$\overline{\text{idx}}\,(c_1) \coloneqq \text{idx}\,(c_1) + 2\eta\,(c_1) - 0.5\ .$$

Based on the other types of external connections, we define (using algorithmic notations) the number of external connections of an



index	-4
adjusted	-2
external	0
tree	0
joint	0
loop4	2
loop2	0
loop0	0
defect	4

(a) (b) (c) (d) (e)

Fig. 9. A simple example of a 3-torus with a single cone. The input is a 20-cone and three tunnel loops (a). A cset is created for the cone. Using alg. 1, an iset is created for the cset. Using alg. 2, an iset chain is created (with a single member), assigning four corners to the iset. According to its index, the iset is even. It is assigned three double loops and four corners. The problem in eqs. (4) to (10) is solved to determine the internal connections of the iset after reducing a loop and adjusting the cone index (section 4.5). The table in (b) shows the results. The table shows for the iset the assigned connections of a single cone of adjusted index -2: two double-loop connections. The defect of the iset (with the reduced index) is 4, which accounts for four corners and turning number 1 of a self-overlapping polygon. Note that due to the loop reduction, this is the same iset case (out of 76 cases) as in fig. 3. The connection graph is shown (c), which includes the reduced loop, and connections are traced over the surface following appendix A. (d) shows the traced seam over the surface. The seam is cut, and the problem in section 6 is solved to determine the metapolygon angles and edge lengths (e). Four corners are marked with squares.

iset as

$$n^{ex} \coloneqq \begin{cases} 2 & I.\text{loop2conn} > 0 \\ I.\text{tree1conn} & I.\text{tree1conn} > 0 \\ 0 & \text{else} \end{cases},$$

and the iset's adjusted "local genus" as

$$\bar{g} \coloneqq I.\text{loop4conn} + \left\lfloor \frac{\sum_{i=0}^{n-1} \overline{\text{idx}}\,(c_i)}{-2} \right\rfloor.$$

In the following ILP, eqs. (4) to (10), we solve for the number of connections of each type for every cset in I. The constraints ensure the defect range, consistency with the genus, and general connectivity expectations such as having a single connected component. Vector constraints are element-wise. A stand-alone constant is treated as a vector of the appropriate size (inferred from the context).

The problem variables are:

- $x^{lp4} \in \mathbb{N}_0^n$ is the number of double-loop connections in each cset.
- $b^{jnt}, b^{tr}, b^{ex} \in \mathbb{Z}_2^n$ indicate if each cset has a joint connection, a tree connection, and all the assigned external connections (respectively).
- $x^{lp0} \in \mathbb{N}_0^n$ is the number of 0-2 connection providers, and $b^{lp2} \in \mathbb{Z}_2^n$ is the number of 0-2 connection receivers.

Each set of constraints is followed by details and explanation. Some of the constraints define auxiliary variables that are not part of the problem output.

The problem:

$$\min_{b,x} \left\|\Delta^c\right\|_\infty \quad \text{subject to} \tag{4}$$

$$\Delta^c \geq -1 \tag{5a}$$

$$1 = \left\|b^{ex}\right\|_1 \tag{5b}$$

$$\bar{g} = \left\|x^{lp4}\right\|_1 + \left\|b^{jnt}\right\|_1/2 \tag{5c}$$

- For a $y \in \mathbb{N}_0^n$, $\left\|y\right\|_1 = \sum_{i=0}^{n-1} y_i$.
- $\Delta^c \in \mathbb{Z}^n$ is the defect of the csets (definition 3.8).
- Equation (5a) bounds the minimal defect as required for monotonicity.
- Equation (5b) sets all the external connections (even if there are zero) to a single cset.
- Equation (5c): $\bar{g}$ equals the number of double loops.

Constraints involving the total number of connections:

$$x^{tot} = b^{tr} + 2b^{lp2} + 3b^{jnt} + 4x^{lp4} + n^{ex}b^{ex} \tag{6a}$$

$$x^{tot} = 4\bar{g} + 2\,(n-1) + n^{ex} \tag{6b}$$

$$\Delta_i^c = 2x_i^{tot} - 4\left(1 - \overline{\text{idx}}\,(c_i)\right), \quad \forall c_i \in I \tag{6c}$$

- Equation (6a) defines $x^{tot} \in \mathbb{N}_0^n$ as the total number of connections in each cset. Each type of connection contributes to a vertex the number of connections that was specified in section 4.2.
- Equation (6b): The graph of cset connections consists of a tree with $n-1$ edges, $\bar{g}$ double-loops (a joint loop can be viewed as a shared double loop with a tree connection), and external connections.
- Equation (6c) is derived from eq. (2).

Constraints related to joint-loop connections:

$$2b^{jtr} \leq b^{jnt} + b^{tr} \tag{7a}$$

$$\left\|b^{jnt}\right\|_1 \leq 2\left\|b^{jtr}\right\|_1 \tag{7b}$$

- Equation (7a) defines $b^{jtr} \in \mathbb{Z}_2^n$, which indicates if a cset has a joint-loop connection and a tree connection.
- Equation (7b) ensures that a joint component (two csets connected by a joint loop) is connected to the rest of the graph—to the other csets (via a tree connection)—where $\left\|b^{jnt}\right\|_1/2$ is the number of joint components.

Constraints related to 0-2 loop connections:

$$b^{lp0} \leq x^{lp0} \leq 4b^{lp0} \tag{8a}$$

$$\left\|x^{lp0}\right\|_1 = \left\|b^{lp2}\right\|_1 \tag{8b}$$

$$b^{lp02} \le b^{lp0} + b^{lp2} \le 2b^{lp02} \tag{8c}$$

$$2b^{tr02} \le b^{tr} + b^{lp02} \tag{8d}$$

$$4\left(1-\beta_0\right) \ge \left|\left\|b^{lp0}\right\|_1 - n + 1\right| \tag{8e}$$

$$\left\|b^{lp0}\right\|_1 \le \left\|b^{tr02}\right\|_1 + 4\beta_0 \tag{8f}$$

$$1 \le x^{lp4} + 4\left(1 - b^{lp0}\right) \tag{8g}$$

- Equation (8a) bounds x^{lp0}. $b^{lp0} \in \mathbb{Z}_2^n$ indicates if a cset c_i is a 0-2 connection provider ($x_i^{lp0} > 0$). Four is a bound on n, which is the maximum number of csets in an iset (alg. 1).
- Equation (8b): the number of 0-2 connection providers equals the number of receivers.
- Equation (8c) defines $b^{lp02} \in \mathbb{Z}_2^n$, which indicates if a cset belongs to a 0-2 component (which is composed of a provider and a receiver).
- Equation (8d) defines $b^{tr02} \in \mathbb{Z}_2^n$, which indicates if a cset belongs to a 0-2 component and also has a tree connection.
- Equations (8e) to (8g) are formulated as Big M constraints (section 3.2).
- Equation (8e) defines a flag $\beta_0 \in \mathbb{Z}_2$, which indicates if the csets in I form a (small) even chain, i.e. are connected to each other only via 0-2 connections.
- If this (small) chain (inside I) is not (all) even, then eq. (8f) ensures that each 0-2 component is connected to the rest of the graph (via a tree connection). $\left\|b^{lp0}\right\|_1$ is the number of 0-2 components.
- Equation (8g): A 0-2 connection provider must have a loop.

If $n > 1$, we then add a constraint for minimum connectivity:

$$1 \le b^{tr} + b^{jnt} + b^{lp0} + b^{lp2} \tag{9}$$

If the first cset is to receive an external joint connection, we then add a constraint to ensure it receives the extra double-loop connection:

$$1 \le x_0^{lp4} \tag{10}$$

After the connection assignment, the isets are connected as described in appendix A.

Notice that throughout the process, the double loops are distributed between cones according to their index. The assignment takes place in the step of assigning external connections (section 4.4) and when adjusting a cone index by reducing loops and assigning them to it (definition 4.7). The loops provide connections to balance the index of a cone to adjust its defect.

Proposition 4.8. *The algorithm that constructs the seam (section 4) handles correctly all cases of negative fields: the connection assignment problem is always feasible, and the traced seam sets the defect of all isets to zero except for the corner isets, which have total defect 4.*

5 POSITIVE CONES

We describe how to construct a seam for fields that also have positive cones. Foundation positive cones (section 5.1) are added to a foundation cset, which adjusts its index, and no further treatment is needed. Non-foundation positive cones (section 5.2) are grouped with negative cones to form balanced csets.

We begin by identifying the foundation set. Let d be the maximum cone degree in a field with n cones. Let $\hat{h} \in \mathbb{N}_0^d$ be the cone histogram, classified by cone degree. Define a vector of weights $w \in \mathbb{N}^d$,

$$w_i \coloneqq \begin{cases} 100 & 1 \le i \le 3 \\ 1 & \text{else} \end{cases}$$

that corresponds to the histogram. We solve an ILP for a histogram $h \in \mathbb{N}_0^d$ of the foundation set:

$$\min_h \quad w^\intercal h \tag{11a}$$

$$\text{s.t.} \quad h \le \hat{h} \tag{11b}$$

$$\sum_{j=1}^{d} h_j \left(1 - \frac{j}{4}\right) = \chi, \quad \forall c_i \in I \tag{11c}$$

We minimize the number of cones, where a larger penalty is given to positives.

Given a histogram, a specific set of cones is picked arbitrarily due to the combinatorial nature of the algorithm, which was explained in the end of section 4.

5.1 Foundation Csets

We partition the foundation set into negative-dominant csets (definition 4.1). We start by creating a cset for each negative cone. Then, we distribute the positive cones between these csets in a greedy way:

- We iterate the positive cones, prioritizing large (positive) index.
- We add a positive cone to a cset with the lowest (negative) index.

Proposition 5.1. *We end up with negative-dominant csets in the foundation set.*

After forming the csets, we connect each cset internally, where positive cones are connected as leaves to the negatives (similar to [Levi 2022, section 7]). We proceed with creating isets for the foundation csets (section 4.1), and connections to a cset are made to its negative cone.

Consequently, in the treatment of the foundation set, the general problem of a field containing positive cones is reduced to the setting of negative fields: a foundation cset with positive cones is treated as a cset with the negative cone only, but with an increased index (which equals the total index of the cset).

5.2 Non-foundation Csets

Forming the csets. After identifying the foundation set, the rest of the cones are divided into non-foundation csets (definition 4.1), using the same procedure that forms the balanced cone sets (definition 3.9) in [Levi 2022, section 6]. A cset is created for each negative cone. Positive cones are added to these csets greedily, prioritizing csets with lower (negative) index. When needed, csets are united to decrease their combined field index to balance a free positive cone.

Internal cset connections. In each cset, the negatives (there could be up to three in a united set) are connected in a chain, and the

positives are connected to the negatives (greedily prioritizing defect balancing) via tree connections.

External cset connections. If it is an even chain of foundation isets (section 4.4), then each non-foundation cset is assigned a 0-2 connection receiver (made of two connections), and an arbitrary foundation cset that owns a loop is assigned a 0-2 connection provider. If the iset chain is odd, then each non-foundation cset is assigned two tree connections. When tracing the connections in appendix A, each of the two connections to a non-foundation cset is made to each end of its negative chain (of one up to three negative cones).

All the non-foundation csets are added to a new (last) iset, which is connected to the seam using the procedure in appendix A.

5.3 An Illustrative Example

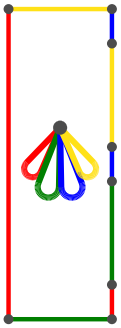
A simple example of a 4-torus with a positive cone is given in fig. 10. The input is a set of 12 cones, $\{2, 6 \times 5, 2 \times 6, 7, 8, 13\}$-cones, and four tunnel loops. A cset is created for each negative cone. After solving eq. (11), nine cones (all negative in this case) are selected as the foundation set with index $\chi = -6$. Four foundation isets are formed using alg. 1. Following section 5.2, the two (negative) non-foundation csets, each with a 5-cone, are united into one cset to balance (the index of) the positive 2-cone, which is added to it. A fifth iset is created to contain the non-foundation (balanced) cset. Using alg. 2, an iset chain is created, assigning four corners and external connections to the isets. According to their index, the first four isets are odd and the fifth is even.

Each of the first two isets is assigned a double loop, an external tree connection, and two corners. Isets 3 and 4 are joined with an external joint-loop connection as an odd pair. Each of them is assigned an external tree connection. Iset 5 is assigned two external tree connections. Following section 4.5, a double loop is (temporarily) reduced from the 13-cone, and its index is adjusted to -0.25. The external joint-loop connection of the odd pair is implemented in the problem by assigning a double-loop for each iset in the pair, and adjusting the index of the first cone of each of these isets by 0.5. The problem in eqs. (4) to (10) is solved to determine the internal connections of the four foundation isets.

The defect of each of the first two isets is 2, which accounts for two corners, and their total defect accounts for the turning number of the polygon. The rest of the isets have defect zero. In iset 4, the first cone receives the external joint connection and shares an ownership of its double loop. It provides 0-2 loop connections to the second and third cones in the iset. One of them is connected with the tunnel loop and the other is connected with the handle loop. The fourth cone in the iset has an internal and an external tree connection.

6 SOLVING FOR THE POLYGON

After cutting the mesh into a disk, we need to specify the edge lengths and angles of the metapolygon. Due to the (non-contractible) loops, unlike in [Levi 2022], a valid angle solution does not necessarily have a corresponding edge-length solution. For example, the polygon of the 2-torus with a single cone in the inset has a valid set of angles, but it is not possible to make all twin-edge lengths equal for this set. Therefore, we need to solve for the angles and edge lengths simultaneously.



Before solving for the polygon, we unite pairs of odd isets that are: (i) corner isets; or (ii) externally connected by a joint-loop connection. The motivation is two-fold: (i) to conveniently obtain a single iset with all four corners at all times; and (ii) to ensure that each iset will have at most two polylines (a sequence of angles) in the polygon. For example, before the union, an external joint connection between a pair of odd isets alternates between copies of at least two cones, each from a different iset, which creates more than two polylines (separate sequences) for each iset. By uniting an odd pair, we unite their polylines as well, making it possible to set the total defect of each one to zero.

After the union, all isets have even index. We will refer to the isets before the union step as odd or even isets and after the union step simply as isets.

Solving for the polygon angles and edge lengths is done via a MILP (which is solved only once). For clarity of expositions, the MILP is described in parts in different sections of the paper. Each section offers additional variables and constraints to the problem (and possibly an adjustment to its objective):

- Section 6.1 is related to polygon angles.
- Section 6.2 is related to polygon edge lengths.
- Section 8 is related to setting loop holonomies.
- Appendix B is related to intersection constraints.
- Appendix D handles the special case of a 1-torus.

For the MILP to be practical, we solve for a small subset of angles at a time; see section 7.

Once we have the polygon, we proceed with the rest of the pipeline as in the sphere case (section 3.1): improving the metapolygon (maximizing the minimum triangulation angle), mapping the interior, and final optimization of the initial map.

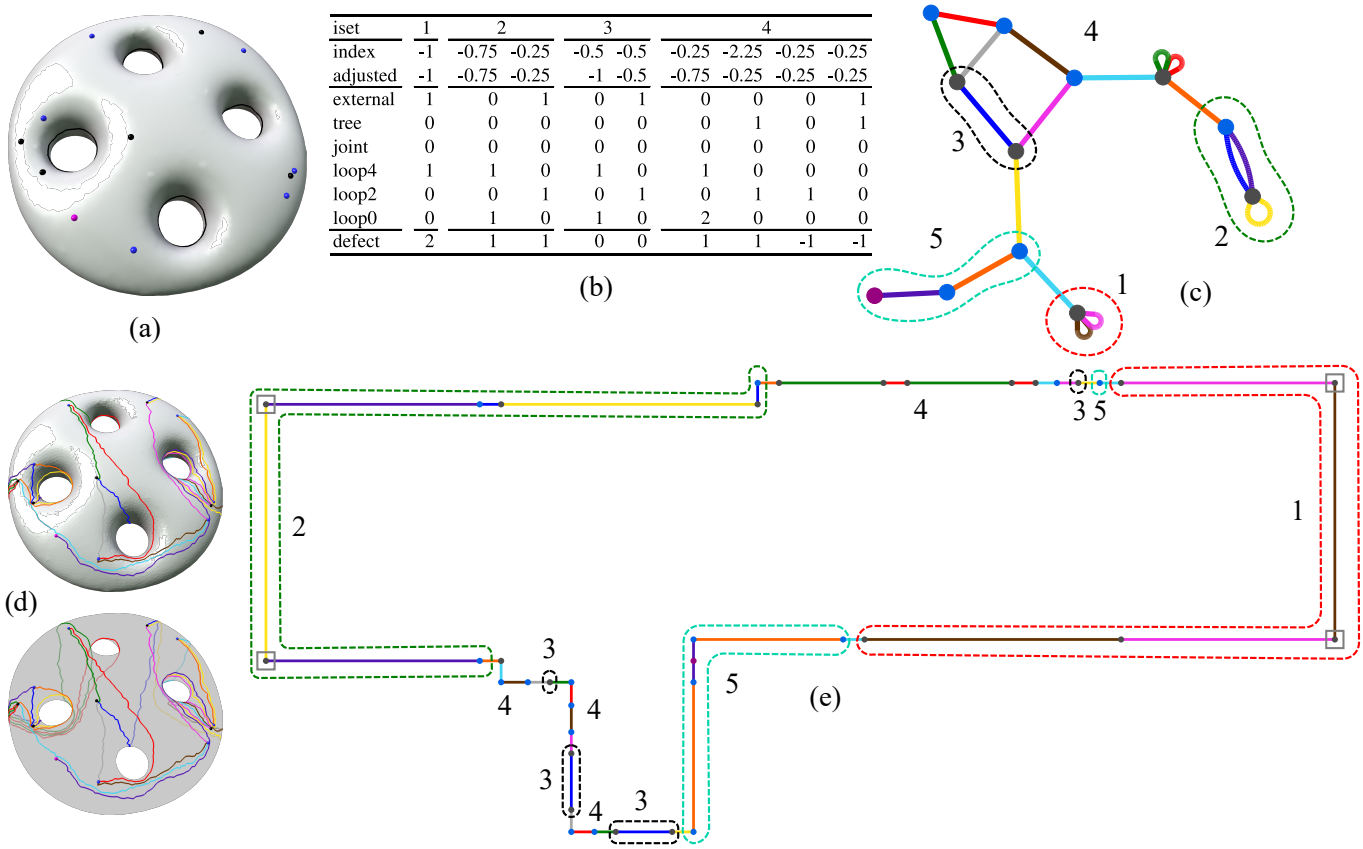
6.1 Polygon Angles

This section defines the initial MILP with variables and constraints that determine the metapolygon angles. The constraints set and restrict the corners to corner isets. They limit the defect of all angle sequences between corners to achieve monotonicity. This follows the same idea of (mostly) alternating left and right turns that was discussed in section 3.1, which is a consequence of limiting the defect. At this part, we are only concerned with feasibility, and the objective is set arbitrarily.

The variables of the problem are $\alpha \in \mathcal{A}^n$, a vector of polygon angles ordered CCW, with $\mathcal{A} \coloneqq \{90^\circ, 180^\circ, 270^\circ\}$. n is the size of the polygon. At the end of section 8, we suggest adding 360° to $\mathcal{A}$ to increase the DOFs (degrees of freedom).

The problem:

$$\begin{aligned} &\min_{\alpha} \ 1 \\ &\text{subject to} \end{aligned} \tag{12}$$



iset	1	2		3		4			
index	-1	-0.75	-0.25	-0.5	-0.5	-0.25	-2.25	-0.25	-0.25
adjusted	-1	-0.75	-0.25	-1	-0.5	-0.75	-0.25	-0.25	-0.25
external	1	0	1	0	1	0	0	0	1
tree	0	0	0	0	0	0	1	0	1
joint	0	0	0	0	0	0	0	0	0
loop4	1	1	0	1	0	1	0	0	0
loop2	0	0	1	0	1	0	1	1	0
loop0	0	1	0	1	0	2	0	0	0
defect	2	1	1	0	0	1	1	-1	-1

Fig. 10. An example of a 4-torus with a positive cone, illustrating concepts. For details, see section 5.3. The input is a set of 12 cones, $\{2, 6\times5, 2\times6, 7, 8, 13\}$-cones, and four tunnel loops (a). (b) the internal connections, determined by solving eqs. (4) to (10). The first row labels the isets. The second and third rows show the index and adjusted index of each cone in the isets. Rows 4-9 show the final connection assignment for each cone. The tenth row shows the defect of each cone. It is calculated from its adjusted index and total number of connections. The connection graph is shown (c), and connections are traced over the surface (d) following appendix A. Each of the isets except for the fourth is circled with a dashed line. The numbers 1–5 denote iset numbers. Internal isets connections are between cones of the same iset. The seam is cut, and the problem in section 6 is solved to determine the metapolygon angles and edge lengths (e). The iset copies are circled with a dashed line as above. Four corners are marked with squares. Meta-edges between corners are monotone. The resulting polygon is simple and hence trivially self-overlapping, which allows for a locally-injective mapping from any disk domain (the cut surface in our case).

$$\bar{\alpha} = A\alpha \tag{13a}$$
$$0 = b_i^{cor}, \quad \forall i \in \left\{ i \in \mathbb{Z}_n \;\middle|\; I_i \notin \mathcal{I}^{cor} \right\} \tag{13b}$$
$$4 = \|b^{cor}\|_1 \tag{13c}$$
$$\alpha \le 90^\circ + \mu_1 \left(1 - b^{cor}\right) \tag{13d}$$

- Equation (13a) constrains the sum of the copy angles of a cone to be equal to the cone angle. $\bar{\alpha} \in \mathbb{R}^m$ is a given column vector of cone angles (from the input). $A \in \mathbb{Z}_2^{m\times n}$ is a binary matrix, where the rows represent the cones, the columns represent the domain copies, and a cell is 1 iff the copy belongs to the cone. We have that $A\alpha$ sums up for each cone all the angles of its copies.
- Equation (13b) limits corner angles to corner isets. $b^{cor} \in \mathbb{Z}_2^n$ indicates if an angle is a corner. I_i refers to the iset that contains α_i. $\mathcal{I}^{cor}$ is the set of all corner isets.
- Equations (13c) and (13d): There are four corner angles that are set to 90°. Equation (13d) is formulated as a Big M constraint (section 3.2) with μ_1 (= 360°).

Sequence sum constraints:

$$s = B\left(180^\circ - \alpha\right) \tag{14a}$$
$$|s| \le 90^\circ + \mu_1 B b^{cor} + \mu_1 \overleftarrow{B}\left(1 - b^{cor}\right) \tag{14b}$$
$$|s| \le \mu_1 B b^{cor} + \mu_1 \left(\overleftarrow{B} + \overrightarrow{B}\right)\left(1 - b^{cor}\right) \tag{14c}$$
$$0 = C\left(180^\circ - \alpha\right), \tag{14d}$$

- Let $L \in \mathbb{Z}_2^{n\times n}$ be a unit triangular matrix (definition 3.15). $B \in \mathbb{Z}_2^{n^2\times n}$ is defined as a column stack of binary matrices of the form $B_i \in \mathbb{Z}_2^{n\times n}$, each equals to some circular shift of L's columns. Since we do not know the corner angles yet, there is a matrix B_i for each negative copy in a corner iset with angle α_i (a potential corner). B_i's columns are a circular shift of L's such that its first

row has its single nonzero cell at column $(i+1) \mod n$, i.e. each sequence (defined in a row of B_i) starts directly after a potential corner angle. In eq. (14a), we have that $s = B\left(180^\circ - \alpha\right)$ equals the sums of angle defects of all sequences between corners that start directly after a corner, for each possible choice of corners.

- Equation (14b) limits the (absolute value of the) sum of a sequence in (a row of) B to 90°. This is similar to [Levi 2022, eq. (2c)]. The constraint is *active* (in terms of a Big M constraint) if the sequence starts directly after a corner and does not contain corners. This ensures monotonicity of polylines between corners. $\overleftarrow{B}$ is a binary matrix derived from B: each of its rows has a single nonzero cell right before the beginning of the sequence.
- Equation (14c) ensures that a meta-edge between corners is straight (the polyline has defect zero). $\overrightarrow{B}$ is a binary matrix derived from B: each of its rows has a single nonzero cell directly after the sequence ends.
- Equation (14d) sets the defect of a polyline of a non-corner iset to zero. C is a binary matrix, where each of its rows corresponds to a sequence of angle defects of an iset polyline.

6.2 Polygon Edge Lengths

The variables and constraints in this section determine the polygon edge lengths, and they are similar to [Levi 2022, eq. (3)]. However, since we solve for the angles as well, the edge directions are unknown. Therefore, we also solve for the edge directions (as auxiliary variables), where we have constraints that tie them to the angle variables.

Variables that are part of the problem output:

- Edge lengths $l \in \mathbb{R}^n$.

Constraints that set edge directions from angles:

$$\mu_2 \left(1 - b^{ang}_{i,k}\right) \geq \left|d_{i+1} - R_{90k^\circ} d_i\right|,\ i \in \mathbb{Z}_{n-1},\ k \in \mathbb{Z}_4 \tag{15a}$$

$$\|b^{ang}_i\|_1 = 1,\ i \in \mathbb{Z}_{n-1} \tag{15b}$$

$$\alpha_i = 180^\circ b^{ang}_{i,0} + 90^\circ b^{ang}_{i,1} + 360^\circ b^{ang}_{i,2} + 270^\circ b^{ang}_{i,3},\ i \in \mathbb{Z}_{n-1} \tag{15c}$$

$$d_0 = (0,1)^\intercal \tag{15d}$$

Equation (15a) determines an edge direction $d_{i+1} \in \mathbb{R}^2$ from a previous edge direction d_i and the angle α_i between them ($\mu_2 = 10$). R_{90k° is a $90k^\circ$ 2D rotation matrix. $b^{ang} \in \mathbb{Z}_2^{n\times 4}$ indicates which of four possible angle values each angle has. Equation (15b) forces one choice per angle. Equation (15c) sets an angle value for α_i according to the choice in b^{ang}_i. Equation (15d) sets an arbitrary direction for the first edge.

Constraints that involve edge vectors and coordinates:

$$v_i = l_i d_i,\ i \in \mathbb{Z}_n \tag{16a}$$

$$1 \leq l_i \leq l_{max},\ i \in \mathbb{Z}_n \tag{16b}$$

$$l_i = l_j,\ \forall (i,j) \in \mathcal{E}_{twin} \tag{16c}$$

$$v_i = x_{i+1} - x_i,\ i \in \mathbb{Z}_n \tag{16d}$$

- Equation (16a) sets the ith edge vector $v_i \in \mathbb{R}^2$ from the direction d_i and an edge length $l_i \in \mathbb{R}$. This is a bilinear constraint; see an equivalent formulation in appendix C.
- Equation (16b) bounds the edge lengths, where $l_{max} \in \mathbb{R}$ is a constant (we used 1000).
- Equation (16c): twin edges have the same length. $\mathcal{E}_{twin} \subset \mathbb{Z}_n \times \mathbb{Z}_n, |\mathcal{E}_{twin}| = \frac{n}{2}$, contains pairs of corresponding twin-edge indices.
- Equation (16d) sets the UV coordinates $x \in \mathbb{R}^{2\times n}$ from edge vectors (and the index i is cyclic).

Proposition 6.1. *For a mesh of genus $g > 1$, there is a solution to the problem in eqs. (12) to (16) with the additional constraints in appendices B.2 and B.3.*

Corollary 6.2. *For a mesh of genus $g > 1$, there exists a quad mesh (extracted from the seamless mapping after mapping the interior into the polygon) of corresponding singularities for any set of cones with total index χ, and each has index < 1.*

7 POLYGON ANGLE SUSPENSION

To make the MILP in section 6 practical, we solve for a small set of variables at a time. We leverage the dummy edge idea in the proof of proposition 6.1 to solve for the angles of one iset at a time while keeping the others suspended.

We begin by solving for the angles of the corner iset. This leaves two polylines that consists of a sequence of angles of isets with defect zero. We suspend these angles by setting them to arbitrary values (and temporarily disable relevant constraints). In each sequence of suspended angles, we set the first two (if there is more than one angle) to 90° and 270°, and the rest of the angles are set to 180°. The first two angles add thickness to the suspended polyline, which allows for a bounding box of arbitrary size (i.e. it provides space and flexibility when the real angles are set).

After solving for the corner angles, we solve for the angles of each balanced iset (defect zero) at the time, except the last one, while:

- Keeping the direction of edges incident to angles of the corner iset fixed.
- Keeping the angles of the other balanced isets suspended.

When solving for the last iset with the non-foundation csets (and no loops), we enable all the isets (set the real angles). Since the foundation isets are already determined, it is possible in this final iteration to solve for angles and lengths separately, as done in [Levi 2022]. Meaning, first solve for the foundation polygon, ensuring consistency of angles and lengths by solving for them simultaneously, and then solve for the non-foundation csets in two steps (angles and then lengths). But in practice the difference in performance was insignificant.

This scheme effectively limits the mixed-integer problem size to about the size of a problem that solves for the angles and edge lengths of a single foundation iset.

8 SETTING LOOP HOLONOMIES

Given a field, the holonomy of a dual non-contractible loop that starts on one side of a seam edge and ends on its other (without crossing the seam for simplicity) can be calculated using parallel transport (based on the cross field). The holonomy can be represented as (field) index, which is equal to the index (or matching

divided by four) of the seam edge up to an integer. Over the (domain) polygon, the same holonomy of the loop can be calculated from the defect of the edge sequence between one twin of the seam edge to the other. Over the surface, this can be viewed as a dual loop the starts at the seam edge from one side (the first twin) and goes along the seam until reaching the other side of the seam edge (the second twin), and it is homotopic to the loop described above. The relation between the defect k of a polygon edge sequence and the index of the corresponding dual loop ℓ over the surface is

$$k = 4\left(1.5 - \mathrm{idx}\left(\ell\right)\right) \ .$$

We use that to measure loop holonomies in terms of defect.

Let $B^h \in \mathbb{Z}^{2g\times n}$ be a matrix indicating angle sequences of the loops and $\Delta^h \in \mathbb{Z}^{2g}$ be the given loop holonomies. To enforce the holonomies, we can add the constraint

$$\Delta^h = B^h\Delta \ ,$$

where $\Delta \coloneqq 2 - \dfrac{\alpha}{90^\circ}$, and α is the vector of polygon angles (section 6.1).

However, enforcing a hard constraint is impractical since:

(1) Our constraints, such as monotonicity, may limit the space of possible holonomies.
(2) If the input holonomies were extracted from a field that was generated arbitrarily, e.g. to promote smoothness [Bommes, Zimmer, et al. 2009], then they may not be feasible. A field is set by a choice of edge matchings (period jumps). Indices of resulting cones will always sum up to χ, and the only required restriction is that cone indices are < 1. Then, except for the special case of a $\{3,5\}$-cone pair on a torus, every such cone set is feasible. Loop holonomies are bounded. Consider the sequence of angles in the polygon between twin edges that a loop crosses. The loop holonomy will be bounded by the angles that can be set in the sequence, which depend on cone indices. Even if this bound is satisfied, it is still an open problem to determine if they are feasible (a locally injective mapping with these holonomies exists).

Therefore, we decided to use a soft constraint instead, where we add to the objective in eq. (12):

$$\lambda_h\left\|\Delta^h - B^h\Delta\right\|_2^2 , \tag{17}$$

which turns the problem into a mixed-integer quadratic program (MIQP). λ_h (we used 10^3) is a weighing constant. All the λ weighing constants are chosen such that there is no competition between the terms in the objective but a clear priority. An alternative implementation is to use a hierarchy of multiple objectives, which is supported, e.g., by [Gurobi 2018]. Thus, if the holonomy case can be handled by the algorithm, then the holonomies would be satisfied exactly.

To allow more degrees of freedom (DOFs), we add 360° to $\mathcal{A}$ (section 6.1). While this causes some edges to overlap, the polygon is still (weakly) simple (up to perturbation).

8.1 Monotonicity Constraint Deactivation

To further increase the DOFs, we allow a restricted amount of monotonicity violation. This involves modifying the constraint in eq. (14b) (adding a term) and replacing it with:

$$|s| \le 90^\circ + \mu_1 B b^{cor} + \mu_1 \overleftarrow{B}\left(1 - b^{cor}\right) + \mu_1 b^{mon} \tag{18a}$$

$$\left\|b^{mon}\right\|_1 \le n_{mon} \tag{18b}$$

along with adding a term to the objective (eq. (12)):

$$\lambda_{mon}\left\|b^{mon}\right\|_1 . \tag{19}$$

$b^{mon} \in \mathbb{Z}_2^{n^2}$ indicates if a sequence constraint was (forcefully) deactivated. λ_{mon} (we used 1) is a weighing constant, and n_{mon} is the maximum number of allowed constraint deactivations.

To keep the violation local, we limit the length of an edge after a monotonicity violation:

$$\left|l_{\overrightarrow{i}+1} - l_{mon}\right| \le \mu_3\left(1 - b^{mon}\right) \ , \tag{20}$$

where $l_{mon} \in \mathbb{R}$ (=2) is a bound on the edge length. $\overrightarrow{i} \in \mathbb{N}_0^{n^2}$ is the index in the polygon of the last element in each sequence. μ_3 is set to l_{max} (=1000). Since the violation is limited, it will occur in the last angle of a sequence: if α_i causes a violation, which is remedied in α_{i+1}, then only the monotonicity of a sequence that ends with α_i will be violated (assuming it is the only violation). There is only one such sequence that ends with α_i that can be active (depending on the choice of the preceding corner, which determines the start of the sequence).

In our experiments we tried with and without the monotonicity constraint deactivation. The price of deactivating the constraints is an increase in the number of intersection constraints, which affects the solver run time. In conjunction with the angle suspension scheme (section 7), we used $n_{mon} = 0$ when solving for the first (corner) and last (non-foundation) isets, and $n_{mon} = 3$ for all the rest. This improved some of the solutions without significantly increasing the number of intersection constraints; see table 1.

9 EVALUATION

We used [Dey et al. 2013] to extract g disjoint loops. Their algorithm uses a random direction. For most models, the tunnels from a single iteration sufficed. However, because their implementation is not foolproof, some models required several runs (starting from a different random direction) and the selection of a blend of tunnels and handles.

To solve the (mixed-)integer problems, we used [Gurobi 2018].

Run time. The experiments were conducted on a laptop. Selecting the foundation set involves solving eq. (11), which takes a fraction of a second. Determining the internal connections of an iset involves the limited size problem in eq. (4), which takes a fraction of a second to solve. Tracing the seam over the surface is done as per [Levi 2022], and it takes between two seconds for a small model (helmet) up to 85 seconds for a larger one (bozbezbozzel). Solving for the polygon was done using the angle suspension scheme (section 7) that solves iteratively for each iset (angles) at a time. Each iteration solves the problem in eq. (12) which, for genus ≥ 2, typically took less than a second for a small polygon (helmet) up to a minute for a larger polygon with a handful of intersection constraints (thai_statue). The final optimization was conducted similar to [Levi 2022], using an external solver [Shtengel et al. 2017], which can take up to several

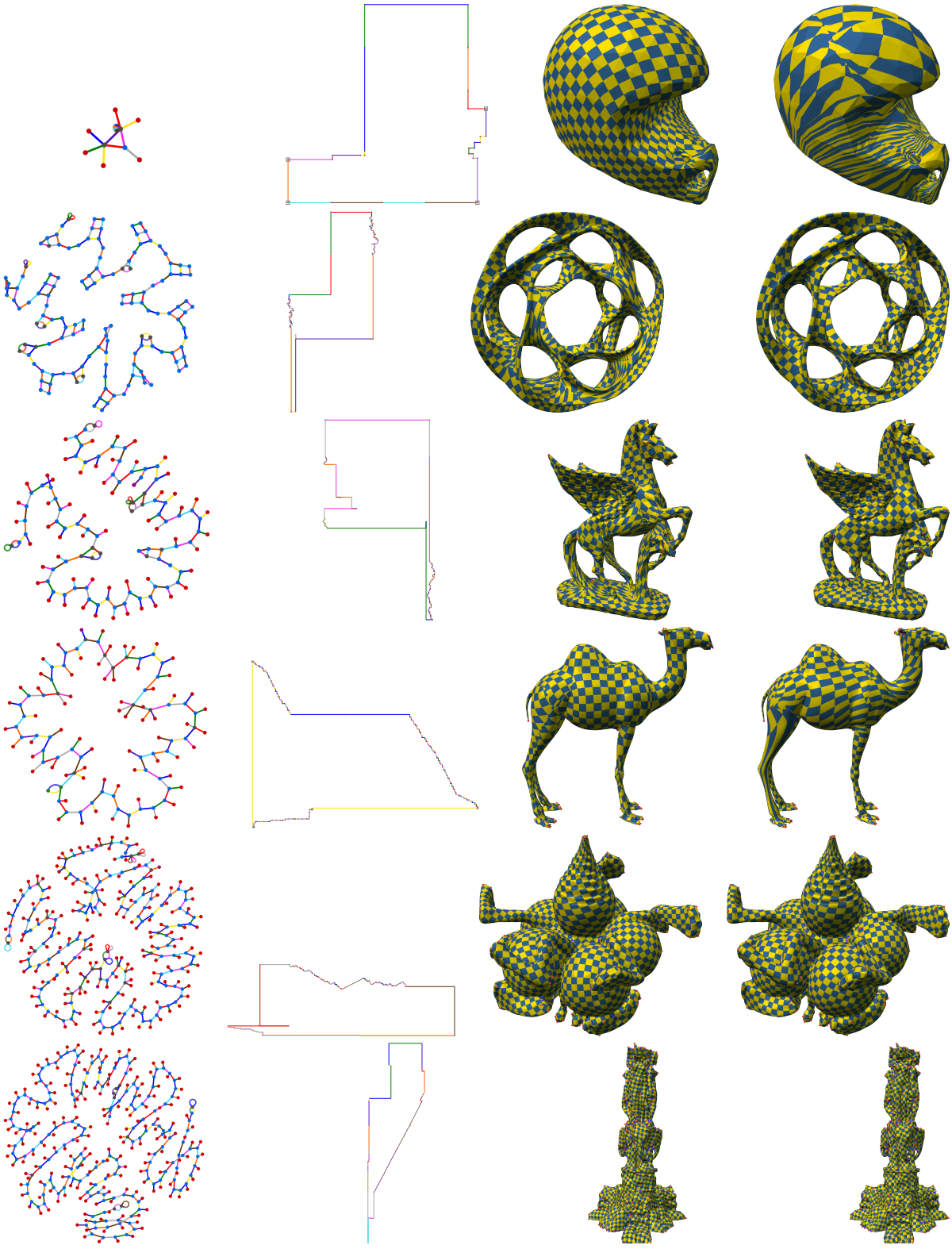


Fig. 11. Models from table 1. From left to right: the seam graph, polygon, final mapping, and [Myles, Pietroni, et al. 2014].

		input				Myles14			ours							
#	model	faces	g	cone	range	faces	energy	added	faces			energy		seam	poly	opti	diff
1	block	4K	3	46	3–5	13K	0.04	2	5K	24K	6K	0.35	0.26	6s	281s	57s	3
2	botijo	82K	5	70	3–5	117K	0.09		86K	180K	85K	0.79	0.44	16s	209s	0.3h	3
3	bozbezbozzel	100K	5	305	2–12	181K	0.13		119K	716K	128K	0.18	0.16	85s	35s	4.3h	5
4	camel	69K	1	121	2–7	111K	∞		75K	208K	72K	0.3	0.3	28s	76s	0.3h	2
5	carter	100K	7	64	3–5	150K	0.19		103K	194K	102K	0.63	0.2	24s	76s	0.5h	1
6	casting_refined	37K	9	119	3–6	67K	0.04		39K	101K	42K	0.33	0.19	19s	562s	11s	3
7	chair	100K	7	98	3–7	162K	∞		110K	271K	104K	1.12	1.12	27s	54s	0.7h	4
8	dancer2	18K	1	53	2–9	42K	0.47	4	20K	57K	20K	0.78	0.78	5s	5s	126s	2
9	dancing_children	100K	8	212	2–7	165K	0.09		120K	464K	112K	0.67	0.41	52s	192s	1.7h	6
10	dragonstand	104K	1	233	3–7	166K	∞		111K	442K	111K	0.46	0.44	43s	175s	1.4h	1
11	elephant	50K	3	118	3–7	83K	0.05		51K	114K	52K	0.23	0.18	14s	24s	456s	3
12	fertility_tri	28K	4	60	3–5	45K	0.04		30K	94K	31K	0.04	0.04	9s	72s	388s	0
13	genus3	13K	3	22	3–5	21K	0.07		15K	32K	14K	0.19	0.07	6s	78s	83s	0
14	greek_sculpture	50K	4	112	1–8	86K	∞		55K	150K	54K	0.23	0.14	16s	86s	674s	2
15	helmet	1K	3	9	3–22	5K	∞		1K	2K	1K	0.65	0.66	2s	15s	3s	2
16	heptoroid	100K	22	140	4–8	168K	0.15		166K	834K	125K	1.76	1.33	62s	35s	325s	10
17	holes3	12K	3	24	3–5	21K	0.05		12K	23K	12K	0.05	0.05	6s	113s	29s	0
18	master_cylinder	100K	3	32	3–5	124K	0.09		101K	170K	101K	0.12	0.12	11s	159s	659s	1
19	neptune0	105K	3	212	3–8	167K	0.08		185K	1.5M	135K	1.67	1.69	62s	49s	10.6h	4
20	oil_pump	100K	4	212	2–6	168K	0.07		109K	407K	110K	0.21	0.15	43s	164s	1.6h	3
21	pegaso	31K	6	131	2–14	63K	0.10		52K	305K	45K	4.1	3.45	15s	106s	0.8h	8
22	rolling_stage	100K	7	52	3–5	138K	0.03		101K	171K	102K	0.29	0.33	19s	178s	0.4h	4
23	seahorse2	100K	8	216	3–20	171K	∞		131K	399K	113K	0.6	0.53	32s	60s	3.0h	12
24	thai_statue	80K	3	366	3–7	160K	0.06		94K	710K	105K	0.15	0.1	39s	155s	6.4h	4
25	twirl	10K	1	16	1–6	17K	0.19	4	11K	16K	11K	0.77	0.77	4s	1s	30s	2

Table 1. Statistics. “g”: genus. “cone”: number of cones. “range”: range of cone quad degree. “energy”: the symmetric Dirichlet energy in eq. (21) after optimization. ∞ indicates a value $> 10^5$, which means that the optimizer could not make progress due to numerical issues caused by nearly collapsed triangles. “added”: the number of added cones compared to the input. In our results, the two columns under “energy” are without and with restricted violation of monotonicity constraints (section 8.1). The three columns under “faces” describe the size of the mesh after tracing the seam, mapping the interior, and simplification. “seam” and “poly”: time (in seconds) to trace the seam and solve for the polygon. “opti”: the time (in seconds or hours) it took for the external solver [Shtengel et al. 2017] to optimize the initial mapping. “diff”: the number of loops whose holonomies are different than the input in the result with the restricted violation.

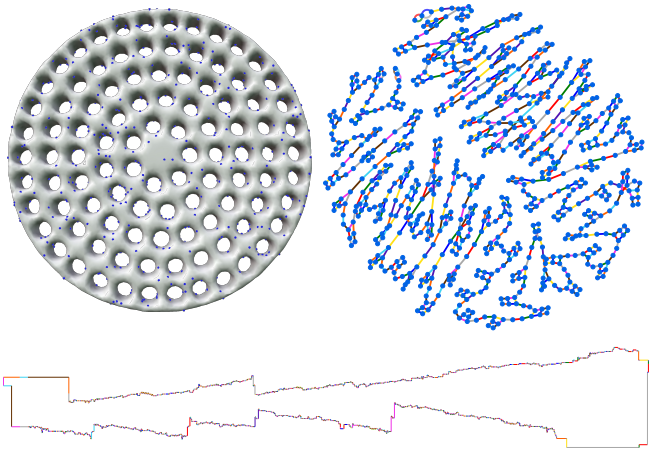


Fig. 12. A stress test on a model of genus 100 with 792 cones.

hours, depending on the model size.

[Zhou et al. 2020]. The method can handle any genus and cone degree ≥ 2, but their experiments were limited to genus ≥ 3. We used the same 20 models that were used in that paper plus five additional models to extend the genus and cone ranges; see table 1 and figs. 1, 11 and 17 to 19. The models are from [Myles, Pietroni, et al. 2014]’s dataset, where the fields were generated using [Bommes, Zimmer, et al. 2009], followed by a post-processing step that joins nearby cones (that, e.g., enforce a higher quad-grid resolution otherwise). Note from table 1 how 1-cones were naturally formed in the twirl and greek_sculpture by this common process.

The results in [Zhou et al. 2020] are provided for the polygon construction only (without testing it with the rest of the pipeline: interior mapping and optimization). Thus, the sum of the “#F init” and “#F pad” columns and the “Time” column in [Zhou et al. 2020, table 1] can be compared to our first column of “faces” and the sum of “seam” and “poly” in table 1. For example, for the thai_statue, their algorithm run time was 7403s and produced an auxiliary quad mesh with 6280K quad faces that a polygon can be extracted from [Levi 2022, appendix H]. This polygon is full, and its shortcomings are discussed in [Levi 2022, appendix G.3]. Moreover, the polygon

sets random loop holonomies (that cannot be controlled). In comparison, our polygon construction took 194sec. It is a metapolygon whose size depends on the number of cones (and their degree). Thus, it allows necessary optimization (increasing the minimal angle in the coarse metapolygon triangulation) [Levi 2022, appendix H] that increases the quality of the interior mapping [Levi 2022, appendix E] that alleviates the issue of nearly collapsed triangles. Our method does not use an auxiliary mesh, and tracing the seam resulted in a mesh with 94K triangles. Another example is the heptoroid in figure 12 in [Zhou et al. 2020], which illustrates their polygon complexity. Compare it with our simple polygon in fig. 11.

[Myles, Pietroni, et al. 2014]. The method traces the cross field over the surface, and it is likely to preserve cones and loop holonomies in the layout. However, this is not guaranteed. For example, their mapping results of the block, dancer2, and twirl contain additional $\{3, 5\}$-cone pairs. This is not only a violation of the theoretical problem at hand, but it also has practical consequences. Using chains of $\{3, 5\}$-cone pairs, [Myles and Zorin 2013] showed that the mapping distortion can be made arbitrarily low. However, these singularities cannot be controlled and may form in unintended places, where the close proximity between cones in a pair would enforce higher quad mesh resolution. Another issue is that the generated mapping contains nearly collapsed triangles. These cause numerical issues that the state-of-the-art optimizer [Shtengel et al. 2017] could not handle, and it ended prematurely; see table 1.

The mapping quality was measured using the symmetric Dirichlet energy [Rabinovich et al. 2017; Smith and Schaefer 2015]:

$$\sigma_1^2 + \frac{1}{\sigma_1^2} + \sigma_2^2 + \frac{1}{\sigma_2^2} - 4\,, \tag{21}$$

where σ_1 and σ_2 are the singular values of the a triangle mapping. Whenever the optimizer succeeded when using a [Myles, Pietroni, et al. 2014]'s mapping as initialization, it converged to (what is likely) the global minimum. This provides a target to compare our result with. In our result, in only three of the models all the loop holonomies are set exactly like the input. Accordingly, for these models the energy is similar to [Myles, Pietroni, et al. 2014], and it is higher for the rest of the models, where some of the holonomies are suboptimal. The model that is affected by this the most and has the highest distortion is the pegaso.

10 LIMITATIONS AND FUTURE WORK

We extended the robust method of [Levi 2022] to a higher genus. The key for robustness is the construction of a small metapolygon, which can be optimized before the interior mapping.

We addressed the problem of prescribing loop holonomies, which can significantly affect mapping distortion. Our solution, however, is limited due to our monotonicity constraints that keep our polygon simple. One direction to improve this limitation is to take advantage of the partitioning of the polygon into polylines (section 7). One may consider polylines that self-intersect (locally), but still have the same total defect, which would increase the DOFs.

The proof of proposition 6.1 shows feasibility for the problem with a core set of constraints. These do not include the bounding box constraints in appendix B.4, which reduce the number of potential intersections. Also, the proof does not include the added DOFs in appendix B.1 nor the acceleration strategy in section 7.

For the 1-torus, we did not prove feasibility for the case of a field that has a 1-cone and no 2-cones (appendix D.5). Thus, while our solution worked in all our experiments, we did not provide guarantees that it is always successful. We note that previous work does not handle 1-cones at all.

All other limitations of the rest of the pipeline that were noted in [Levi 2022, section 11] still stand:

- The method relies on interior mapping. Current interior mapping methods produce a large mesh with excess refinement, which hinders performance of optimization methods.
- Sharp features, which are crucial in some application (e.g. CAD), were not addressed. A possible direction is noticing that our polygon edges are axis-aligned. If the seam passes through features curves, then alignment is guaranteed.
- The method now handles a surface of any genus but addresses only closed surfaces. In a surface with boundary, the geodesic curvature of the boundary can be prescribed. One direction to consider is using a double cover, where a surface is duplicated and glued at the boundary. This creates a closed surface with extra cones whose index determines the geodesic curvature of the boundary. The closed surface can be parameterized using the current method and cut in half.

We leave these directions to explore in future work.

REFERENCES

Noam Aigerman and Yaron Lipman. 2015. "Orbifold tutte embeddings." *ACM Trans. Graph.*, 34, 6, 190–1.

David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013. "Integer-grid maps for reliable quad meshing." *ACM Transactions on Graphics (TOG)*, 32, 4, 98.

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013. "Quad-Mesh Generation and Processing: A Survey." *CGF*, 32, 6, 51–76.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. "Mixed-integer Quadrangulation." *ACM Trans. Graph.*, 28, 3, 77:1–77:10.

Marcel Campen, David Bommes, and Leif Kobbelt. 2015. "Quantized Global Parametrization." *ACM Trans. Graph.*, 34, 6, 192:1–192:12.

Marcel Campen, Ryan Capouellez, Hanxiao Shen, Leyi Zhu, Daniele Panozzo, and Denis Zorin. 2021. "Efficient and robust discrete conformal equivalence with boundary." *ACM Transactions on Graphics*, 40, 6, 1–16.

Marcel Campen, Hanxiao Shen, Jiaran Zhou, and Denis Zorin. 2019. "Seamless parametrization with arbitrary cones for arbitrary genus." *ACM Transactions on Graphics*, 39, 1, 1–19.

Marcel Campen and Denis Zorin. 2017. "Similarity maps and field-guided T-splines: a perfect couple." *ACM Transactions on Graphics (TOG)*, 36, 4, 1–16.

Der-San Chen, Robert G Batson, and Yu Dang. 2010. *Applied integer programming: modeling and solution.* John Wiley & Sons.

Wei Chen, Xiaopeng Zheng, Jingyao Ke, Na Lei, Zhongxuan Luo, and Xianfeng Gu. 2019. "Quadrilateral mesh generation I: Metric based method." *Computer Methods in Applied Mechanics and Engineering*, 356, 652–668.

Edward Chien, Zohar Levi, and Ofir Weber. 2016. "Bounded Distortion Parametrization in the Space of Metrics." *ACM Trans. Graph.*, 35, 6, 215:1–215:16.

Tamal K Dey, Fengtao Fan, and Yusu Wang. 2013. "An efficient computation of handle and tunnel loops via Reeb graphs." *ACM Transactions on Graphics*, 32, 4, 1–10.

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. "Integrable PolyVector Fields." *ACM Trans. Graph.*, 34, 4, 38:1–38:12.

Xianzhong Fang, Hujun Bao, Yiying Tong, Mathieu Desbrun, and Jin Huang. 2018. "Quadrangulation Through Morse-parameterization Hybridization." *ACM Trans. Graph.*, 37, 4, 92:1–92:15.

Michael Floater. 2003. "One-to-one piecewise linear mappings over triangulations." *Mathematics of Computation*, 72, 242, 685–696.

Mark Gillespie, Boris Springborn, and Keenan Crane. 2021. "Discrete conformal equivalence of polyhedral surfaces." *ACM Transactions on Graphics*, 40, 4.

Steven J Gortler, Craig Gotsman, and Dylan Thurston. 2006. "Discrete one-forms on meshes and applications to 3D mesh parameterization." *Computer Aided Geometric Design*, 23, 2, 83–112.
Branko Grunbaum. 1969. "Planar maps with prescribed types of vertices and faces." *Mathematika*, 16, 1, 28–36.
Gurobi. 2018. *Gurobi Optimizer Reference Manual*. (2018). http://www.gurobi.com.
Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. "Instant field-aligned meshes." *ACM Trans. Graph.*, 34, 6, 189–1.
E. Jucovič and M. Trenkler. 1973. "A theorem on the structure of cell–decompositions of orientable 2–manifolds." *Mathematika*, 20, 1, 63–82.
F. Kälberer, M. Nieser, and K. Polthier. 2007. "QuadCover: Surface Parameterization using Branched Coverings." *Computer Graphics Forum*, 26, 3, 375–384.
Zohar Levi. 2021. "Direct Seamless Parametrization." *ACM Transactions on Graphics*, 40, 1, 1–14.
Zohar Levi. 2022. "Seamless Parametrization of Spheres with Controlled Singularities." In: *Computer Graphics Forum* 1. Vol. 41, 57–68.
Zohar Levi and Denis Zorin. 2014. "Strict minimizers for geometric optimization." *ACM Transactions on Graphics (TOG)*, 33, 6, 185.
Yaron Lipman. 2012. "Bounded distortion mapping spaces for triangular meshes." *ACM Transactions on Graphics (TOG)*, 31, 4, 108.
Feng Luo. 2004. "Combinatorial Yamabe flow on surfaces." *Communications in Contemporary Mathematics*, 6, 05, 765–780.
Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. "Robust Field-aligned Global Parametrization." *ACM Trans. Graph.*, 33, 4, Article 135, 135:1–135:14.
Ashish Myles and Denis Zorin. 2013. "Controlled-distortion constrained global parametrization." *TOG*, 32, 4, 105.
Ashish Myles and Denis Zorin. 2012. "Global parametrization by incremental flattening." *TOG*, 31, 4, 109.
Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. "Scalable Locally Injective Mappings." *ACM Trans. Graph.*, 36, 4.
N. Ray, W.C. Li, B. Lévy, A. Sheffer, and P. Alliez. 2006. "Periodic global parameterization." *ACM Trans. Graph.*, 25, 4, 1460–1485.
Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. 2009. "Geometry-aware direction field processing." *ACM Trans. Graph.*, 29, 1, 1–11.
Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. "N-symmetry direction field design." *ACM Transactions on Graphics*, 27, 2, 10.
Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. 2019. "Progressive embedding." *ACM Transactions on Graphics*, 38, 4, 32.
Hanxiao Shen, Leyi Zhu, Ryan Capouellez, Daniele Panozzo, Marcel Campen, and Denis Zorin. 2022. "Which cross fields can be quadrangulated? global parameterization from prescribed holonomy signatures." *ACM Transactions on Graphics (TOG)*, 41, 4, 1–12.
Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. "Geometric Optimization via Composite Majorization." *ACM Trans. Graph.*, 36, 4.
Jason Smith and Scott Schaefer. 2015. "Bijective Parameterization with Free Boundaries." *ACM Trans. Graph.*, 34, 4, 70:1–70:9.
Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. "Conformal equivalence of triangle meshes." *TOG*, 27, 3, 77.
Jian Sun, Tianqi Wu, Xianfeng Gu, and Feng Luo. 2015. "Discrete conformal deformation: algorithm and experiments." *SIAM Journal on Imaging Sciences*, 8, 3, 1421–1456.
Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. 2011. "Simple quad domains for field aligned mesh parametrization." In: *Proceedings of SIGGRAPH Asia*, 1–12.
W.T. Tutte. 1963. "How to draw a graph." *Proc. London Math. Soc*, 13, 3, 743–768.
Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. "Directional Field Synthesis, Design, and Processing." *CGF*, 35, 2, 545–572.
Ofir Weber and Denis Zorin. 2014. "Locally injective parametrization with arbitrary fixed boundaries." *TOG*, 33, 4, 75.
Jiaran Zhou, Changhe Tu, Denis Zorin, and Marcel Campen. 2020. "Combinatorial Construction of Seamless Parameter Domains." In: *Computer Graphics Forum* 2. Vol. 39, 179–190.

A PAIRING AND TRACING CONNECTIONS

Connections are made by tracing the seam over the mesh, following the refinement rules in [Levi 2022, appendix C]. Tracing a connection is done by finding a (shortest) path over the mesh edges while avoiding intersection with other parts of the seam.

When connections are assigned, only connection ends are assigned to cones without explicitly pairing them (e.g. four cones were assigned a tree connection each, but we have not specified yet which pairs of cones the two tree connection paths connect), which we specify in this section. Given the assigned internal connections, we connect the csets in an iset following the five steps below. In the steps, we connect two csets only if they belong to different (seam) graph components.

(1) Connect joint loops. Create a list of csets with joint loop connections. Sort the list first by the number of tree connections, and then by the number of joint loop connections, using a stable sort. Iterate the list, connecting a cset from its front with a cset from its back. Each pair of connected csets is removed from the list.
(2) Connect double loops.
(3) Apply 0-2 loop connections. Create two lists. The first, a list of providers, which is sorted by the number of provided 0-2 connections. The second, a list of receivers, which is sorted first by the number of tree connections and then by the number of provided 0-2 connections, using a stable sort. Iterate over the list of providers in a reverse order. If a provider has a tree connection, then it is matched with a receiver from the beginning of the receivers list and otherwise from its end. A connected pair of provider and receiver are removed from the corresponding lists. If the provider provides two connections, then it is inserted back to the beginning of the provider list.
(4) Connect tree connections. Create a spanning tree between the cset graph components. Start from a seed cset with a tree connection. Connect csets which are not in the tree, prioritizing csets with more than one tree connection.
(5) Connect reduced double loops.

After connecting each iset internally, we connect the isets to each other in a chain. Since the joint-loop connections were already made when constructing the iset chain, only 0-2 loop and tree connections need to be made, and the order of the isets in the chain can be arbitrary.

A.1 Seam Edge Order Around a Cone

When connecting a negative-dominant cset, we are mindful of the seam edge order around its negative cone v. This is used in proving proposition 6.1. Joint-loop, 0-2 loop (receiver), and tree connections are performed such that double-loop connections and internal tree connections to positive cones are on one side of the seam. That is, after cutting the seam, the copies of v are divided between two polylines. One polyline will have a single copy due to two external edges going from v, and the other polyline will have the remainder of the copies. For example, if v is externally connected via two tree connections, it will then have these two edges one after the other, when considering the order of seam edges around v. Similarly, for a 0-2 loop-receiver connection, the two edges are one after the other when considering the order of seam edges around the incident (receiver) vertex. It is also possible to ensure the correct order in a joint connection by keeping the external connection to the right of the meta-half-edge that the two loops share.

Levi [2022] concluded that maintaining a specific seam-edge order around a vertex is not necessary for a non-foundation balanced cset c (which is added to an existing foundation polygon). Nevertheless,

we maintain an edge order as described above for the 1-torus case. There could be up to three negative cones in a balanced cset, where a cset with more than one negative cone is termed in [Levi 2022, section 6] a united cset. For genus > 1, the external connections to c are tree connections. For genus 1, the external connections to c are via a generalization of the 0-2 connection (appendix D). In both cases, the requirement is to keep all the positive cones to one side of the seam.

B INTERSECTION CONSTRAINTS

The monotonicity constraints do not guarantee intersection-free polygon but significantly reduce the number of potential intersections to a handful (if any), which need to be resolved. [Levi 2022, appendix K] offers intersection(-free) constraints that are mainly necessary when solving for the foundation polygon, which we use in a similar way. In this section, we adjust and extend these constraints.

We consider the angle suspension scheme (section 7). If there are intersections, most of them occur in the first iteration when solving for the corner iset angles. In subsequent iterations, we solve for the angles of two monotone polylines of a balanced iset (defect zero), which usually do not add intersections, while keeping the edge directions of the corner iset fixed. Similar to [Levi 2022], we encourage longer edges incident to the four corners (appendix B.3), so polylines of a balanced set appear as small features that are unlikely to globally intersect (they cannot intersect locally due to the monotonicity constraints).

Except for the final iset of the non-foundation csets, an iset has a limited size and hence a limited number of angle variables. Except for the first two turns, the rest of a suspended polyline is straight. A straight polyline is efficiently treated in [Levi 2022, appendix K] as a single segment. The result is that in every iteration (except the final one), the polygon has small number of turns, which directly affects the number of potential intersection constraints. Moreover, the intersection constraint set is emptied every new iset iteration (in the angle suspension algorithm).

B.1 Allowing 360° Angles

To allow 360° angles (section 8), we need to allow overlapping edges when checking for segment intersection. Given a sequence of angles that are all 180° except for one which is 360°, we ignore intersections between all edges incident to the angle sequence.

B.2 Specific Edge Directions

Each time intersections are detected in the result, we solve eq. (12) again (for the same angles) with additional intersection constraints. Since we solve simultaneously for angles and lengths, unlike [Levi 2022], edge directions may change when solving again. Therefore, an intersection constraint between two edges needs to be specific to given edge directions.

We use the notations, variables, and definitions in [Levi 2022, appendix K]. Given two intersecting edges, the matrix $A \in \mathbb{R}^{2\times 2}$ consists of the edge directions. Similarly, we define the matrix $\bar{A} \coloneqq \begin{bmatrix} d_i & -d_j \end{bmatrix}$, where $d_i, d_j \in \mathbb{R}^2$ are the (unknown) directions of these edges in the polygon that we are solving for in eq. (15a). We add a condition (using the Big M method with $\mu_5 = 10$) to the last constraint in [Levi 2022, appendix K] ($b_1 + b_2 + b_3 + b_4 \le 3$) that depends upon the two pairs of vectors being equal (eq. (22a)), accompanied by additional constraints:

$$\|b_i\|_1 \le 3 + \mu_5 \left(1 - b^{same}\right) \tag{22a}$$

$$D = A - \bar{A} \tag{22b}$$

$$\epsilon_1 \le \operatorname{vec} D + \mu_5 \left(b^{cell} + b^{neg}\right) \tag{22c}$$

$$\epsilon_1 \le -\operatorname{vec} D + \mu_5 \left(b^{cell} + 1 - b^{neg}\right) \tag{22d}$$

$$|\operatorname{vec} D| \le \epsilon_1 + \mu_5 \left(1 - b^{cell}\right) \tag{22e}$$

$$4b^{same} \le \left\|b^{cell}\right\|_1 \le 3 + b^{same} \tag{22f}$$

where $D \in \mathbb{R}^{2\times 2}$ is the difference between the two matrices. $\operatorname{vec}(\cdot)$ gives the column stack of a matrix. $b^{cell} \in \mathbb{Z}_2^4$ matches the dimension of $\operatorname{vec} A$, and b_i^{cell} indicates if the ith cell of $\operatorname{vec} A$ and $\operatorname{vec} \bar{A}$ is equal. $b^{same} \in \mathbb{Z}_2$ indicates if all four cells are equal ($A = \bar{A}$). $\epsilon_1 \in \mathbb{R}$ (=0.1) is a threshold for two cell values being equal. $b^{neg} \in \mathbb{Z}_2^4$ indicates the sign of each cell of $\operatorname{vec} D$.

B.3 Longer Corner Edges

We add constraints to encourage longer edges incident to the four corners. Define $b^{corE} \in \mathbb{Z}_2^{\frac{n}{2}}$ that indicates for each pair of twin edges in $\mathcal{E}_{twin}$ if (at least) one of the twins is incident to a corner. For the kth pair $(i, j) \in \mathcal{E}_{twin}$, the constraints that define the indicator b_k^{corE} are:

$$b_k^{corE} \le b_i^{cor} + b_{i+1}^{cor} + b_j^{cor} + b_{j+1}^{cor} \tag{23a}$$

$$b_k^{corE} \ge b_i^{cor} \tag{23b}$$

$$b_k^{corE} \ge b_{i+1}^{cor} \tag{23c}$$

$$b_k^{corE} \ge b_j^{cor} \tag{23d}$$

$$b_k^{corE} \ge b_{j+1}^{cor} \tag{23e}$$

We add soft constraints to encourage longer corner edges (only the length of one twin needs to be set due to eq. (16c))

$$\left|l_i - l_{long}\right| \le t + \mu_3 \left(1 - b_i^{cor}\right),\ \forall (i, j) \in \mathcal{E}_{twin} \tag{24a}$$

$$|l_i - 1| \le t + \mu_3 b_i^{cor},\ \forall (i, j) \in \mathcal{E}_{twin} \tag{24b}$$

along with a term to the objective (eq. (12)):

$$\lambda_t t\,, \tag{25}$$

where λ_t (=10^{-3}) is a weighing constant. l_{long} (= $\frac{n}{4}$, clamped to $[10, 100]$) is a constant that determines the preferred length of long edges.

These soft constraints are enabled in the first iteration when solving for the corner iset (section 7), and else we encourage the edge lengths to remain similar to the lengths l^0 from the previous iteration (see [Levi 2022, eq. (3a)]):

$$\left|l - l^0\right| \le t\,. \tag{26}$$

B.4 Bounding Box Constraints

We take advantage of the (coarse) rectangular shape of the metapolygon (section 4.3) and add special intersection constraints to prevent two opposing meta-edges in the rectangle from intersecting each other, which reduce the number of potential intersections (contributing to a faster solution). The constraints are added after solving the first iset angles and determining the corners.

Let σ^{right} be the set of vertex indices on the right meta-edge of the rectangle: from the bottom-right corner to the top-right corner (included). Similarly, define $\sigma^{\cdot}$ for the other three rectangle meta-edges. Also, define the complement sets $\bar{\sigma}^{\cdot} \coloneqq \mathbb{Z}_n \setminus \sigma^{\cdot}$. Determine the bounds on the relevant coordinate (U or V) of each edge, and set opposing edges ϵ_2 (=10) apart:

$$m_{right} \leq x_{\sigma^{right},U} \tag{27a}$$
$$m_{top} \leq x_{\sigma^{top},V} \tag{27b}$$
$$m_{left} \leq x_{\sigma^{left},U} \tag{27c}$$
$$m_{bottom} \leq x_{\sigma^{bottom},V} \tag{27d}$$
$$m_{right} \geq \epsilon_2 + m_{left} \tag{27e}$$
$$m_{top} \geq \epsilon_2 + m_{bottom} \tag{27f}$$

Set a pair of opposing rectangle edges apart from the rest of the polygon:

$$x_{\bar{\sigma}^{right},U} \leq m_{right} - \epsilon_2 + \mu_6 \left(1 - b_{horiz}\right) \tag{28a}$$
$$m_{left} \leq x_{\bar{\sigma}^{left},U} - \epsilon_2 + \mu_6 \left(1 - b_{horiz}\right) \tag{28b}$$
$$x_{\bar{\sigma}^{top},V} \leq m_{top} - \epsilon_2 + \mu_6 b_{horiz} \tag{28c}$$
$$m_{bottom} \leq x_{\bar{\sigma}^{bottom},V} - \epsilon_2 + \mu_6 b_{horiz} \tag{28d}$$

where $b_{horiz} \in \mathbb{Z}_2$ chooses between the horizontal and vertical pairs. We set μ_6 to 10^4.

Seamless Parametrization with Cone and Partial Loop Control—Supplement

ZOHAR LEVI, Victoria University of Wellington, New Zealand

C BILINEAR CONSTRAINTS

For the ith edge, the bilinear constraint in Equation (16a) is equivalently formulated as:

$$1 \le d_{i,U} + \left(1 - b_{i,0}^{dir}\right) \tag{29a}$$
$$d_{i,U} \le -1 + \left(1 - b_{i,1}^{dir}\right) \tag{29b}$$
$$1 \le d_{i,V} + \left(1 - b_{i,2}^{dir}\right) \tag{29c}$$
$$d_{i,V} \le -1 + \left(1 - b_{i,3}^{dir}\right) \tag{29d}$$
$$\|b_i^{dir}\|_1 = 1 \tag{29e}$$
$$\left|v_{i,U}\right| \le \mu_3 \left(b_{i,0}^{dir} + b_{i,1}^{dir}\right) \tag{29f}$$
$$l_i \le v_{i,U} + \mu_3 \left(1 - b_{i,0}^{dir}\right) \tag{29g}$$
$$v_{i,U} \le -l_i + \mu_3 \left(1 - b_{i,1}^{dir}\right) \tag{29h}$$
$$\left|v_{i,V}\right| \le \mu_3 \left(b_{i,2}^{dir} + b_{i,3}^{dir}\right) \tag{29i}$$
$$l_i \le v_{i,V} + \mu_3 \left(1 - b_{i,2}^{dir}\right) \tag{29j}$$
$$v_{i,V} \le -l_i + \mu_3 \left(1 - b_{i,3}^{dir}\right) \tag{29k}$$
$$|v_i| \le (l_i, l_i)^\mathsf{T} \tag{29l}$$

In eq. (15), $d_i = \begin{bmatrix} d_{i,U} \\ d_{i,V} \end{bmatrix}$ is set to one out of four directions. Equations (29a) to (29e) set the indicator $b_i^{dir} \in \mathbb{Z}_2^4$, which decides one state out of four according to d_i. Equations (29f) to (29l) set the vector of the ith edge, $v_i = \begin{bmatrix} v_{i,U} \\ v_{i,V} \end{bmatrix} \in \mathbb{R}^2$. Its direction is determined by b_i^{dir}, and its length is l_i. μ_3 is set to l_{max} (=1000).

D 1-TORUS

In this section, we treat the special case of a 1-torus. We start with the seam construction:

- Pick arbitrary regular vertices u and v to comprise the foundation set.
- Connect the single double loop to u. Mark its four copies as corner angles.
- Connect u to v via a 0-2 connection, using the tunnel loop. We denote the angles of the two copies of v as α_{v_a} and α_{v_b}.
- Connect u to each balanced cset via a 0-2 connection, using the handle loop. In case the cset has more than one negative cone (a united cset—see [Levi 2022, section 6]), then the 0-2 connection is generalized to pass through the cset's negative chain (connecting to both of its ends).

Author's address: Zohar Levi, Victoria University of Wellington, New Zealand.

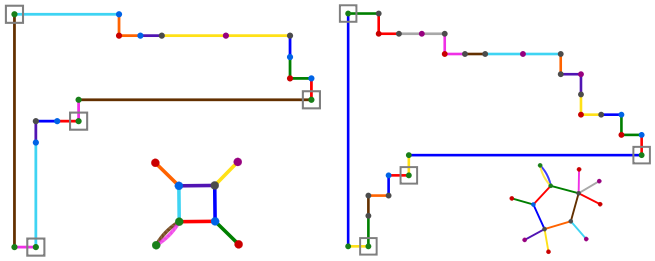


Fig. 13. Two gun-shape polygons.

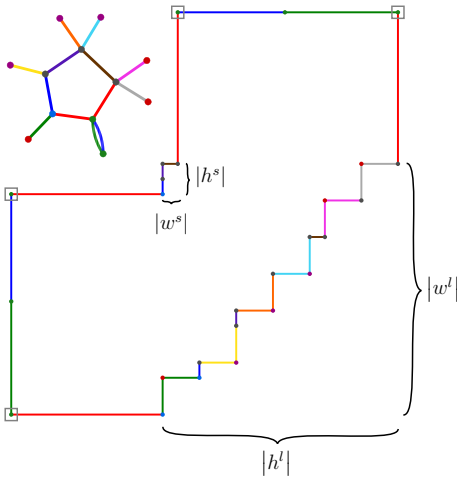


Fig. 14. An L-shape polygon.

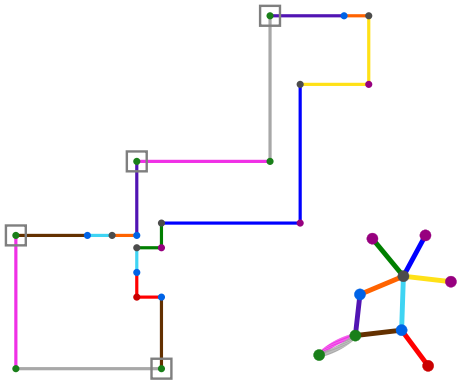


Fig. 15. A stair-shape polygon.

Maintaining the seam edge order around a vertex (appendix A.1), the four meta-edges of the polygon between the corner angles in a CCW order are:

(1) A polyline with α_{v_a}.
(2) A short polyline that consists of a single copy of each negative cone. Let $\alpha^s \in \mathbb{R}^k$ be the (CCW) sequence of the polyline angles.
(3) A polyline with α_{v_b}.
(4) A long polyline that consists of all the rest of the negative copies and all the positive copies. Let $\alpha^l \in \mathbb{R}^m$ be the (CCW) sequence of the polyline angles.

Depending on the field, instead of a (coarse) rectangle (section 4.3), we construct a polygon in one out of three possible (coarse) shapes:

(i) If the field contains a 2-cone, then we construct a *gun-shape*; see fig. 13.
(ii) Else, if the field contains a united cset, then we construct a *stair-shape*; see fig. 15.
(iii) Else, we construct an *L-shape*; see fig. 14.

When solving for the polygon, we add a specific set of constraints (to the problem in section 6) for each shape.

Let $B^l \in \mathbb{Z}_2^{m\times m}$ and $B^s \in \mathbb{Z}_2^{k\times k}$ be unit triangular matrices (definition 3.15). Define the polyline defects:

$$\Delta^l \coloneqq 2 - \frac{\alpha^l}{90^\circ}$$
$$\Delta^s \coloneqq 2 - \frac{\alpha^s}{90^\circ}$$

Define sequence defects in each polyline:

$$\delta^l \coloneqq B^l \Delta^l$$
$$\delta^s \coloneqq B^s \Delta^s$$

D.1 Gun-Shape Constraints

$$0 \le \delta^l \tag{30a}$$
$$1 = \delta^l_\beta \tag{30b}$$
$$90^\circ = \alpha_{v_a} \tag{30c}$$
$$-1 = \delta^s_k \tag{30d}$$
$$0 \ge \delta^s \tag{30e}$$

where

- Equation (30a) restricts α^l from having a right turn with respect to the initial direction (of the sequence): restricts all sequences that start at α^l_0 from turning right, i.e. the total defect of a sequence is ≥ 0.
- Equation (30b) forces the sequence $\alpha^l_0, \dots, \alpha^l_\beta$ to perform a *big* left turn (the total defect of the sequence is 1). β is the index in α^l of the angle of an arbitrary 2-cone on the long polyline.
- Equation (30c) sets the angle of one of the two copies of v. The other one is set by eq. (13a).
- Equation (30d): α^s performs a *big* right turn (the total defect of the sequence is -1).
- Equation (30e) restricts α^s from having a left turn with respect to the initial direction.

D.2 Common Constraints for L-shape and Stair-Shape

We describe the constraints that are the same for both shapes. Similar to eq. (30d):

$$-1 = \delta^s_k \tag{31}$$

Classify turns (left, straight, right) of each sequence on the long polyline:

$$\left|\delta^l - 1\right| \le \mu_4 \left(1 - b^{l\leftarrow}\right) \tag{32a}$$
$$\left|\delta^l\right| \le \mu_4 \left(1 - b^{l\uparrow}\right) \tag{32b}$$
$$\left|\delta^l + 1\right| \le \mu_4 \left(1 - b^{l\rightarrow}\right) \tag{32c}$$
$$1 = b^{l\leftarrow} + b^{l\uparrow} + b^{l\rightarrow} \tag{32d}$$

where $b^{l\leftarrow}, b^{l\uparrow}, b^{l\rightarrow} \in \mathbb{Z}_2^m$ indicate if the sequence $\alpha^l_0, \dots, \alpha^l_i$ performs the turn (total defect) that $b^{l\cdot}_i$ represents. μ_4 was set to 2.

Let $\widehat{\delta^s} \coloneqq B^s_{1:k-1,1:k-1}\Delta^s_{1:k-1}$ be sequence defects of the short polyline without the last angle. Classify turns on the short polyline:

$$\left|\widehat{\delta^s} - 1\right| \le \mu_4 \left(1 - b^{s\leftarrow}\right) \tag{33a}$$
$$\left|\widehat{\delta^s}\right| \le \mu_4 \left(1 - b^{s\uparrow}\right) \tag{33b}$$
$$\left|\widehat{\delta^s} + 1\right| \le \mu_4 \left(1 - b^{s\rightarrow}\right) \tag{33c}$$
$$1 = b^{s\leftarrow} + b^{s\uparrow} + b^{s\rightarrow} \tag{33d}$$

Define the (signed) "width" and "height" of the polylines in terms of turns:

$$w^l \coloneqq \left\|b^{l\rightarrow}\right\|_1 - \left\|b^{l\leftarrow}\right\|_1$$
$$h^l \coloneqq \left\|b^{l\uparrow}\right\|_1$$
$$w^s \coloneqq \left\|b^{s\rightarrow}\right\|_1 - \left\|b^{s\leftarrow}\right\|_1$$
$$h^s \coloneqq \left\|b^{s\uparrow}\right\|_1$$

D.3 Additional L-Shape Constraints

$$180^\circ = \alpha_{v_a} \tag{34a}$$
$$h^l - w^s = -w^l - h^s \tag{34b}$$
$$w^l \le 0 \tag{34c}$$
$$b^{l\rightarrow} = 0 \tag{34d}$$

D.4 Additional Stair-Shape Constraints

$$270^\circ = \alpha_{v_a} \tag{35a}$$
$$w^l + w^s = h^l - h^s \tag{35b}$$
$$\tilde{b}^{l\rightarrow} = 0 \tag{35c}$$

D.5 Feasibility

Proposition D.1. *If there is a 2-cone in the field, then a gun-shape polygon is feasible.*

Proposition D.2. *If the only positives in the field are 3-cones (the setting of [Grunbaum 1969]), then an L-shape polygon is feasible.*

If there is a 1-cone in the field, then we use an L-shape or a stair-shape, depending if there is a united cset. We did not prove feasibility for this case, but the algorithm correctly handled all 1575 fields with the following restrictions:

- The maximum cone degree is 12.
- There are at most four cones of the same degree.
- The total index of the negative cones ≥ -3.

The only two failures were: (i) a field without cones and (ii) a field with a pair of 3-5-cones (where a quad mesh with these singularities does not exist). Note that previous work did not handle 1-cones at all.

For the L-shape and gun-shape, we can solve separately for the angles and the edge lengths, as done in [Levi 2022].

E PROOFS

Proposition 4.8. *The algorithm that constructs the seam (section 4) handles correctly all cases of negative fields: the connection assignment problem is always feasible, and the traced seam sets the defect of all isets to zero except for the corner isets, which have total defect 4.*

Proof. A g-torus has $2g$ loops and index $\chi = 2-2g$ (Poincaré–Hopf theorem). A double loop that is assigned to an even iset or an odd pair (definition 4.6) balances (sets to zero) index amount of -2, as explained in section 4.3 (it provides four connections, each is worth index 0.5). On the one hand, the index of both an even iset and an odd pair is even, and on the other hand, the amount of index that a double loop balances is -2 (a divider of an even number). Therefore, the balancing of each iset's index is perfect (not partial or fractional). Since only $g-1$ double loops are required to balance the index of all isets, it leaves an excess of a single double loop. This double loop is assigned to corner isets, giving excess defect 4, which accounts for turning number 1 of a self-overlapping polygon.

In an (all-)even iset chain, each iset I is assigned $-\operatorname{idx}(I)/2$ double loops to balance its index. According to eq. (2), a balanced iset (or a regular vertex with index zero) requires two connections to balance its defect (set it to zero). Each iset has at least one loop, which can serve as a 0-2 loop connection provider (providing two connections), and an even chain that is based on 0-2 loop connections is created. Except for the first iset, each iset in the chain is a 0-2 loop connection receiver. That is, it receives the two necessary connections that are needed to balance its defect. We still have a double loop left, which we assign to the first iset that is designated as corner iset. Now, this iset has two connections more than it requires to balance its defect, which becomes four, which is also the total defect of the chain, as required for the turning number 1 of the polygon.

For a chain with odd isets, except for the two odd corner isets, the rest of the odd isets are paired (always possible since the total index χ is even). We assign double loops to odd pairs and even isets to balance their index as before (where each odd pair receives at least one double loop that is used for a joint loop connection). The rest of the loops are divided between the two odd corner isets such that the defect (and not index) of each one is zero (for example, an iset with index -1 that receives four connections). Finally, we connect all the even isets, odd pairs, and corner isets in a chain using tree connections, where each corner iset is at one end of the chain. Each even iset and odd pair receives two connections, which balances their defect. Each corner iset receives a single connection, which sets its defect to two. The total chain defect is four, as required.

See section 4.3.1 for simple examples of both chain cases.

This concludes the proof for isets with a single cone. Next, we consider the general case, where an iset can have up to four cones. External connections remain unchanged, and we need to prove that the internal connections in an iset are handled correctly.

Consider the input to the problem in eq. (4). Alg. 1 creates seven types of isets, each has four csets at most. After loop reduction, there are eight types of csets ($\overline{\text{idx}} \in \{-0.25k \mid k = 1, \ldots, 8\}$) that can comprise an iset.

If it is an even iset, then there are three cases of external connections: the iset can be the first in an even iset chain, in the middle of an even chain, or in the middle of an odd chain.

If it is an odd iset, then there are two cases of external connections: the iset can be either at the end of a chain or in the middle.

We end up with 76 cases. The problem in eq. (4) does not need to be correct for a more general case, and it was simplified to treat only these 76 cases. We tested to confirm that all of them are handled correctly. That is, for each case of iset, the internal connection assignment problem is feasible, the defect is set as required, and the connection graph contains a single connected component. Adding back reduced double loops to an iset with adjusted index (and reverting its index) does not affect its defect. On top of verifying the connection assignment for all cases, we also verified that the 5-step pairing procedure in appendix A produces a correct result.

To summarize, the assignment of internal and external connections is feasible, creates a single connected component, and produces the required defect according to assigned corners in all cases. □

Proposition 5.1. *We end up with negative-dominant csets in the foundation set.*

Proof. Assume towards contradiction that in the end of the process that creates foundation csets (section 5), there exists a cset c with non-negative index.

Consider the iteration of adding a positive cone p to a cset c with the lowest index, where before the iteration all the csets were negative-dominant, and adding p made c non-negative. We have that before the iteration:

$$\operatorname{idx}(p) \geq -\operatorname{idx}(c) \ .$$

It cannot be that $\operatorname{idx}(p) = -\operatorname{idx}(c)$ since then $\operatorname{idx}(c \cup \{p\})$ would be a balanced set, which can be removed from the foundation set without influencing its index, contradicting the optimality of eq. (11). Consider the cases of the index of p:

- $\operatorname{idx}(p) = 0.25$. Then, $\operatorname{idx}(c) \geq 0$, which contradicts negative-dominant.
- $\operatorname{idx}(p) = 0.5$. Then, $\operatorname{idx}(c) = -0.25$. Since the total index (χ) is integer, there must be another cset c' with index -0.25 (and there

cannot be another positive cone with index 0.25, which would create a balanced set). But then, $c \cup c' \cup \{p\}$ would be a balanced set—contradiction.
- idx $(p) = 0.75$. Then, all the options to reach an integer χ would result in creating a balanced set—contradiction.

□

Proposition 6.1. *For a mesh of genus $g > 1$, there is a solution to the problem in eqs.* (12) *to* (16) *with the additional constraints in appendices B.2 and B.3.*

Proof. Consider the foundation polygon. In appendix A.1, we specified how the seam connects the isets such that after cutting it, there are two polylines. One polyline has one copy of a negative cone, and the second polyline has all the rest of its copies.

From the 76 types of odd and even isets, 31 are corner isets, and the rest are balanced (defect zero). The minimizer of eq. (4) has maximal cset defect of four. It is achieved in the case of a corner iset in an even chain with a single cset. A cset ends up with defect larger than one only if it belongs to a corner iset.

From eq. (2), we have for a vertex v in a balanced iset (defect zero)

$$\Delta(v) = 2n - \frac{\alpha}{90^\circ} \le 1$$
$$90^\circ \left(2 - \frac{1}{n}\right) \le \frac{\alpha}{n}.$$

That is, the average copy angle is $> 90^\circ$ if $n > 1$. On the other hand, for a negative cone v, the minimal angle is 450°, and the average copy angle is $> 90^\circ$ if $n < 5$. In conclusion, the average copy angle is $> 90^\circ$, and therefore there must be at least one copy with angle $> 90^\circ$. We set the angle of the single copy in a polyline to 180°. The result is that a balanced iset (defect zero) has one polyline that consists of single copies (each from a different cone) with 180° and a second polyline with a sequence of all the other copies. Since a balanced iset has defect zero and the first polyline has defect zero, the second polyline also has defect zero.

There are 34 types of odd isets, half of them are corner isets and the other half are in an external joint-loop connection pair (there are 17 odd iset types with different csets before designating them as corner or in an odd pair). There are 42 types of even isets, third of them are corner isets. After the step of uniting odd isets, the even iset types remain the same. There are $2\binom{17}{2} = 272$ types of paired odd isets (as corners or as joint-loop pairs). Since the number of iset types after the iset union step is still relatively small, it is possible to test all the corresponding polylines for the feasibility of a polygon that consists of a corner iset and a single balanced iset (defect zero).

Given such a polygon, it is possible to add another balanced iset (defect zero) since its two polylines can be set to have defect zero (keeping meta-edges between corners monotone and straight). This addition can be viewed as replacing the single balanced iset with two balanced isets. Since the polygon is feasible for any balanced iset, the bounding box of a polyline of the balanced iset (and its connections to the rest of the polygon) is general and can accommodate two consecutive iset polylines instead. This can be generalized to any number of balanced isets. This concept is similar to a polygon accommodating a dummy edge [Levi 2022, proof of theorem 5], which serves as a stand-in for polylines with similar first and last edge directions (which are straight in this case due to defect zero of the polyline).

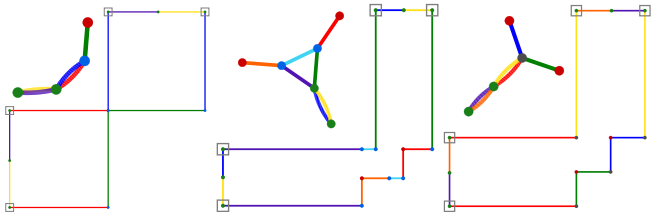


Fig. 16. L-shape polygons, corresponding to the fields (left to right): $\{3, 5\}$-cones, $\{3, 5, 3, 5\}$-cones, and $\{3, 3, 6\}$-cones.

Adding reduced double loops to a polygon can be done by setting all the copy angles of the vertex that owns them to 180° (defect zero). □

Proposition D.1. *If there is a 2-cone in the field, then a gun-shape polygon is feasible.*

Proof. Denote the two edges incident to the regular copy v_a as e_1 and e_2 (CCW). For example, in fig. 13 (left), there are the brown and pink edges. e_1 comprises the left side of the polygon, and we set it to an arbitrary length to match the right side. The bottom side of the polygon contains e_2, which can vary in length arbitrarily without affecting the polygon consistency. The top side of the polygon contains the edge that is incident to the 2-cone, and it can also vary in length arbitrarily. These two degrees of freedom (length of two edges) enable setting the same length for the top and bottom sides of the polygon, ensuring polygon consistency.

The constraints eq. (30a) and eq. (30e) are not necessary, and they serve only to create a nicer gun shape. □

Proposition D.2. *If the only positives in the field are 3-cones (the setting of [Grunbaum 1969]), then an L-shape polygon is feasible.*

Proof. Consider fig. 16. In the polygon with the single $\{3, 5\}$-cone pair, the positive copy coincides with the negative copy on the opposing edge. There are no DOFs in this case to avoid that. The figure shows two additional constructions. One adds another $\{3, 5\}$-cone pair, and the other groups two 3-cones with a 6-cone. These can be generalized to any number and type of balanced csets with 3-cones. □

REFERENCES

Branko Grunbaum. 1969. "Planar maps with prescribed types of vertices and faces." *Mathematika*, 16, 1, 28–36.

Zohar Levi. 2022. "Seamless Parametrization of Spheres with Controlled Singularities." In: *Computer Graphics Forum* 1. Vol. 41, 57–68.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. "Robust Field-aligned Global Parametrization." *ACM Trans. Graph.*, 33, 4, Article 135, 135:1–135:14.

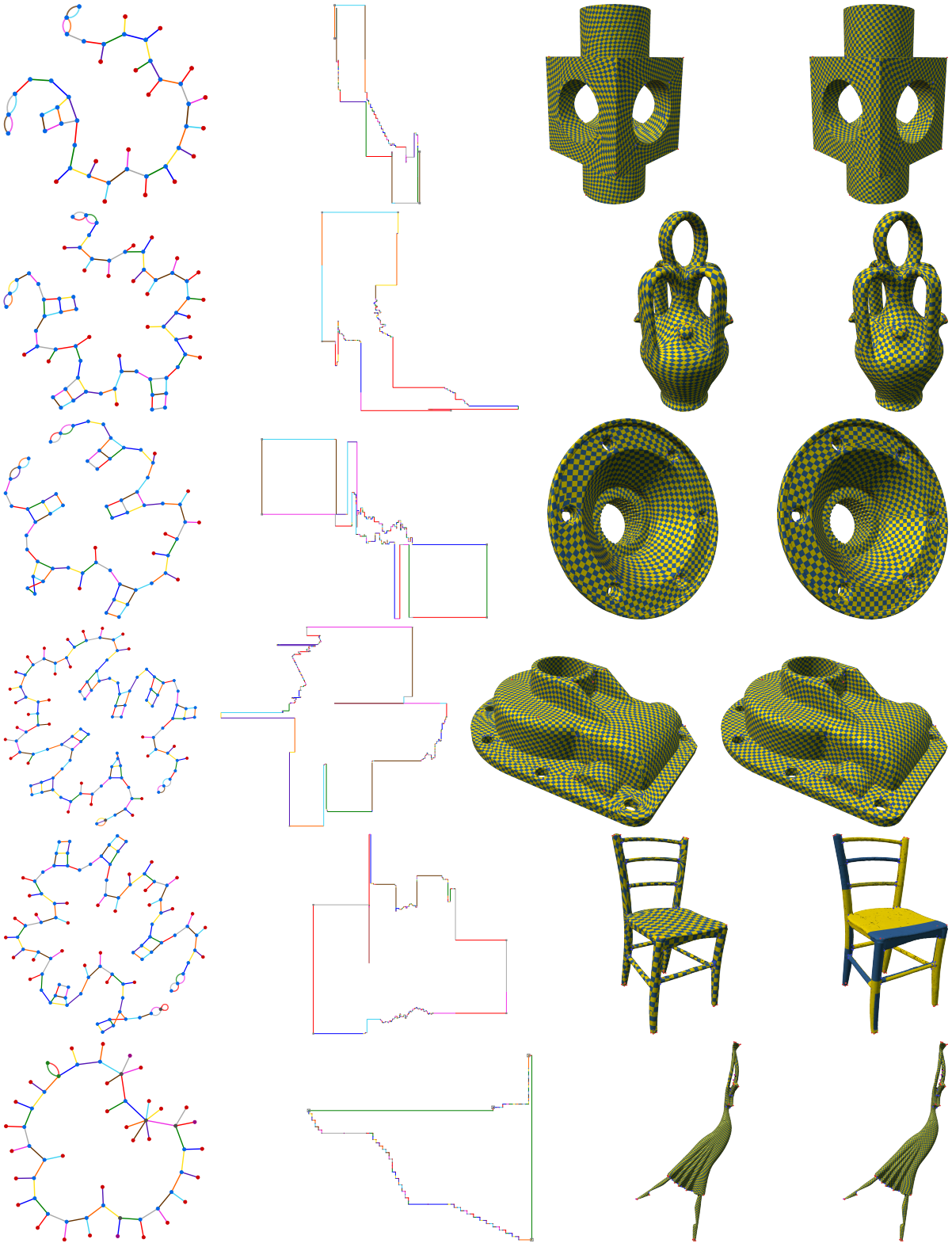


Fig. 17. Models from table 1. From left to right: the seam graph, polygon, final mapping, and [Myles et al. 2014].

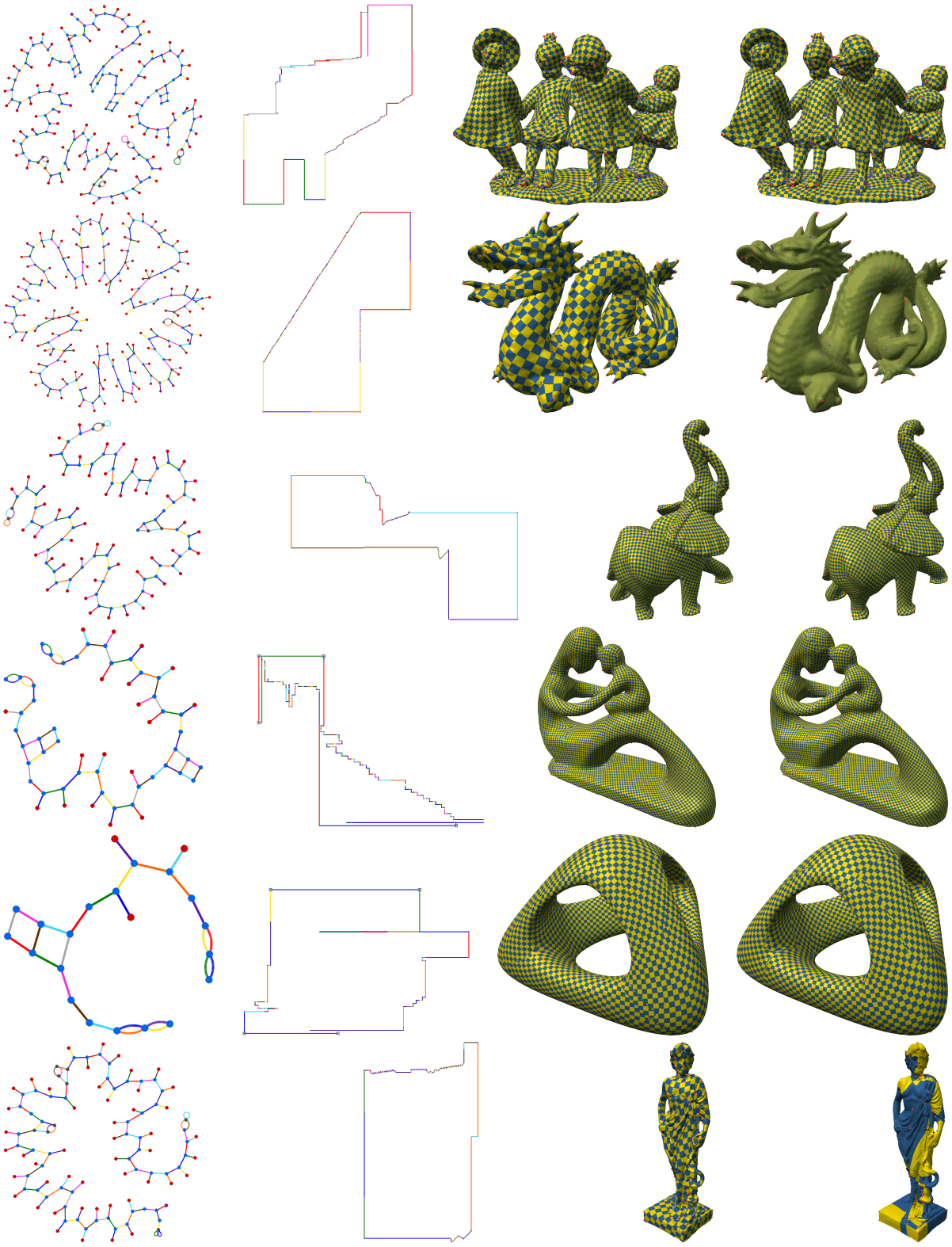


Fig. 18. Models from table 1. From left to right: the seam graph, polygon, final mapping, and [Myles et al. 2014].

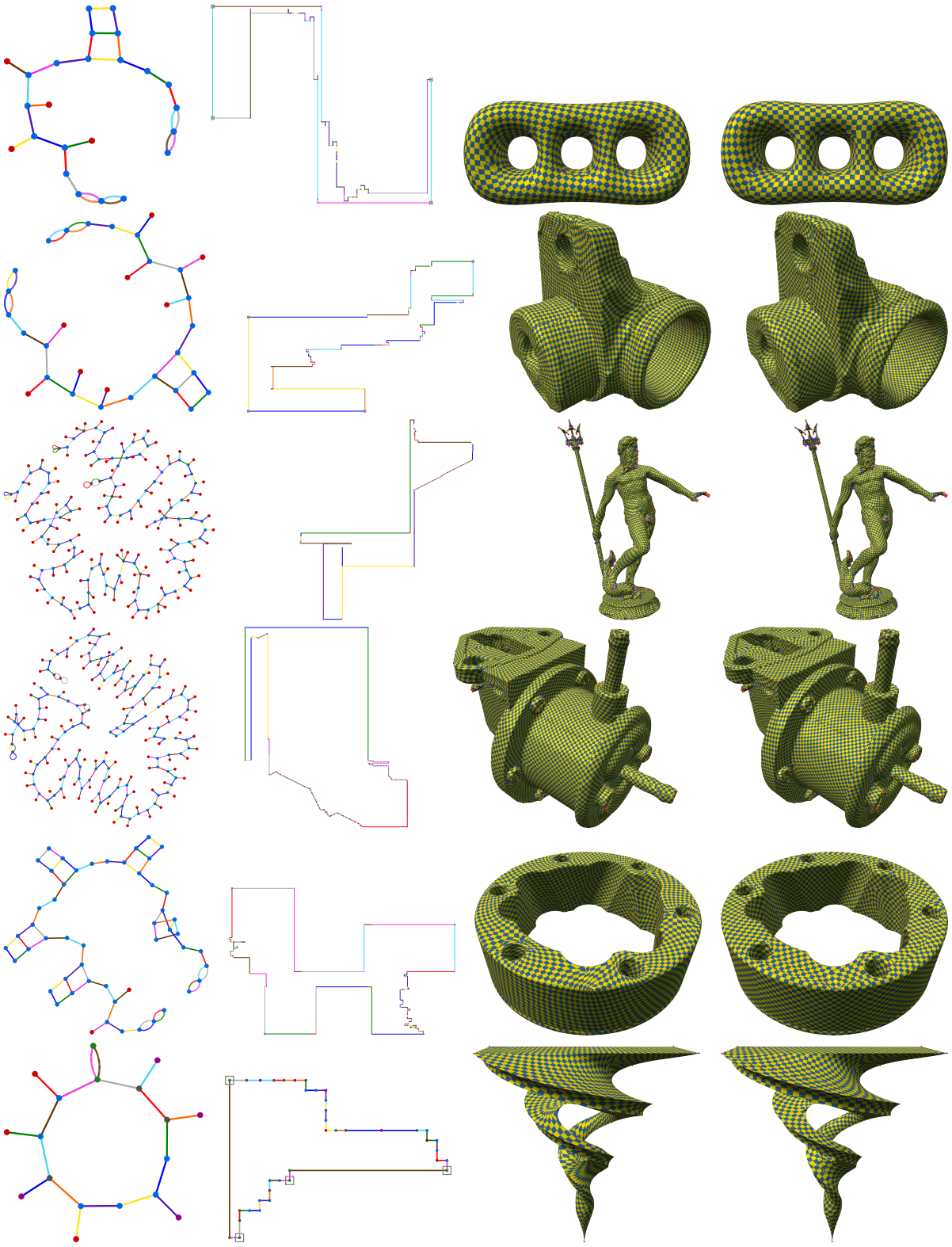


Fig. 19. Models from table 1. From left to right: the seam graph, polygon, final mapping, and [Myles et al. 2014].