DOI: 10.1111/cgf.70421
EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Embedding Optimization of Layouts via Distortion Minimization

A. Heuschling [1] I. Lim [1] L. Kobbelt [1]

[1] RWTH Aachen University, Germany

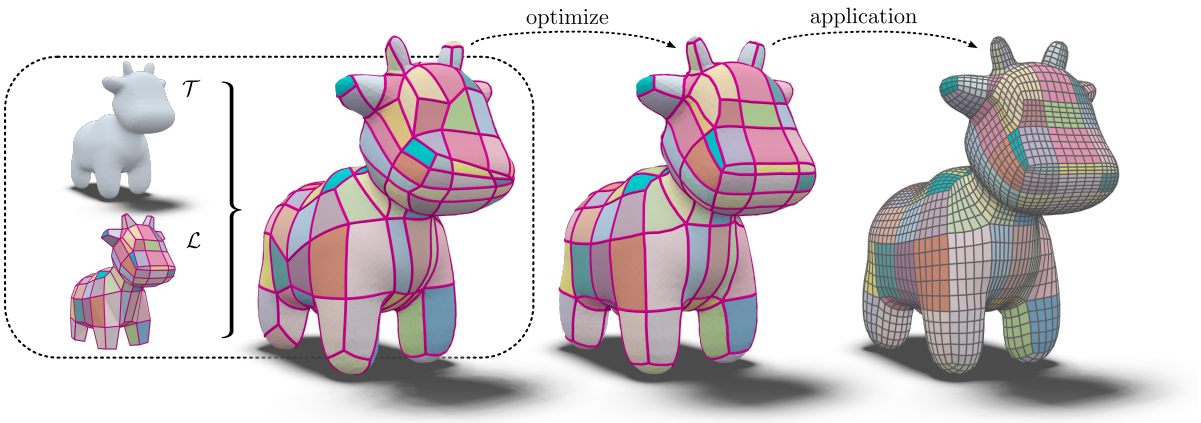


Figure 1: *We propose a method for optimizing the embedding of a given layout. As input, we expect a target surface $\mathcal{T}$, a layout connectivity $\mathcal{L}$, and an initial embedding of $\mathcal{L}$ in $\mathcal{T}$. Our method strictly preserves the connectivity prescribed by $\mathcal{L}$ throughout the optimization process. By repositioning layout nodes, it improves the geometric quality of the embedding from which downstream applications such as quad meshing can directly benefit.*

Abstract

Given an embedding of a layout in the surface of a target mesh, we consider the problem of optimizing the embedding geometrically. Layout embeddings partition the surface into multiple disk-like patches, making them particularly useful for parametrization and remeshing tasks, such as quad-remeshing, since these problems can then be solved on simpler subdomains. Existing methods can either not guarantee to maintain patch connectivity, limiting downstream applications, or are specialized for quad layout optimization, relying on principal curvature information. We propose a framework that balances per-patch distortion minimization with strict connectivity control through an explicit representation. By inserting additional nodes along layout arcs, they can be embedded as piecewise geodesic curves on the surface. This sampling of arcs provides additional flexibility where required, enabling joint optimization of both node positions and arc embeddings. Our representation naturally supports a multi-resolution workflow: optimization on coarse meshes can be prolongated to high-resolution inputs. We demonstrate its effectiveness in applications requiring connectivity-preserving, low-distortion surface layouts.
Code will be available at `https://github.com/7-AlexH/layout-embedding-optimization`.

CCS Concepts

• ***Computing methodologies*** → ***Mesh models; Mesh geometry models;*** *Shape modeling;*

1 Introduction

Layout generation, *i.e.*, the process of partitioning a surface into a coarse network of disjoint disk-topology patches, is a key ingredient in many applications such as parametrization, quad meshing, or shape correspondences. As such, significant effort has been poured into the fully- or semi-automatic generation and optimization of layouts for surfaces.

While the precise role of layouts may differ depending on the application, their main objective is to capture the structure of a complex surface and dividing it into smaller, more manageable parts.

Although there is no universally accepted definition of a good layout embedding, commonly desired properties include embedded arcs that follow principal curvature directions and, most importantly, near-developable patches that yield low-distortion uv-maps.

In the context of surface parametrization, *e.g.*, for texturing, layouts specify the cuts of the curved surface. The placement of these cuts has a significant impact on the quality (*i.e.*, the distortion) of the final parametrization. Typically, the connectivity and geometric embedding are constructed from scratch. A prominent example is the approach by Sharp and Crane [SC18], where the layout connectivity evolves freely during optimization while minimizing parametrization distortion.

While such freedom in connectivity can be advantageous for parametrization, it limits applicability in tasks like quad meshing or shape correspondence via joint layout, where explicit control over the layout connectivity is essential. This highlights the need for approaches that balance geometric quality with structural control, depending on the target application.

In quad layout generation, the initial connectivity is typically established by connecting singularities in a guiding direction field and then quantizing the resulting connections. Geometric optimization is subsequently performed in the parameter domain over the layout *arcs*, *i.e.*, the *uv*-isolines. The positions of quad mesh singularities, corresponding to layout *nodes*, are usually moved only a posteriori through simple smoothing [LCK21b]. A more sophisticated alternative was introduced by [CK14b], who propose an alternating optimization scheme between a global parametrization and the placement of singularities. Their method is tailored to quad layout optimization, as it explicitly leverages alignment with principal curvature directions without explicitly considering the shape of patches.

We aim to fill the gap and propose a method for embedding optimization of layouts driven by a patch-based objective function, in the spirit of Sharp and Crane [SC18], while retaining explicit control over layout connectivity. In contrast to Campen and Kobbelt [CK14b], all degrees of freedom in our formulation live directly on the surface, enabling simultaneous optimization of node and arc embeddings. By representing layout arcs as *piecewise* geodesics, as in Noma *et al.* [NSS*24], we control the flexibility of arcs and allow alignment with curved regions.

Our main contributions are:

- an explicit representation for layout embeddings (Section 3.2), enabling connectivity maintenance throughout optimization,
- an auto-differentiable evaluation procedure for a layout embedding state (Section 4),
- a patch-based objective function penalizing parametrization distortion that is differentiable w.r.t. boundary shape parameters (Section 5), ensuring patch-shape-aware gradients,
- an optimization strategy for handling the non-convex energy landscape of the patch-aware objective (Section 6).

In Section 7, we compare our method to [SC18] and [LCK21b], and demonstrate its effectiveness on downstream applications, *i.e.*, quad meshing and shape correspondence.

2 Related Work

Research on *layout generation and embedding* connects to several neighboring areas, in particular *cone and cut placement* in surface parametrization, and the large body of work on *parametrization*. We review representative approaches in each category that are most relevant to our setting.

Layout Generation. A wide range of methods address layout generation on surfaces. Depending on the application, layouts may be triangular [SAPH04; SCBK20] or quadrilateral. Quad layouts can be obtained through T-mesh quantization [LCK21a; LCK21b], without quantization [CDH*24; PPM*16], or guided by auxiliary structures such as skeletons [PBS22; ULP*15]. Other approaches generate polygonal-patch layouts [PNA*21], while interactive tools allow user-driven design [TPSS13; CK14a]. For a detailed survey of quad layout techniques, see [Cam17]. In general, these methods simultaneously determine the combinatorial structure of layouts and compute a geometric embedding. The resulting layout embeddings can serve as input to our method.

Layout Embedding and Optimization. In applications such as compatible quad remeshing or inter-surface mapping, a prescribed layout must be embedded into a target surface. Greedy approaches [PSS01] have been explored, as well as a branch-and-bound algorithm focusing on the topology of the embedding [BSK21]. Once the connectivity is fixed, research has focused on optimizing the geometry of the embedding, commonly via global parametrization [TPP*11; LCK21b]. Singularities are generally fixed at this stage and only smoothed afterwards, with the exception of [CK14b] and [LNTC24]. In [CK14b] both the singularity placement and parametrization are optimized for curvature alignment. As the method relies on curvature alignment, it is only suitable for quad layout optimization and difficult to apply in the presence of noise. [LNTC24] proposed a framework for optimizing layout embeddings using objectives based on the geodesic lengths of embedded arcs. We follow a similar embedding approach for layout arcs. While their method effectively reduces patch deformation, it tends to distribute the remaining deformation uniformly across all patches. In contrast, our approach allows modification of the parameter domain, which can be interpreted as changing the rest shape. Additionally, their method requires patches to be approximately intrinsically flat.

Cone and Cut Placement. Conformal parametrizations preserve angles but not areas, motivating numerous approaches for cone singularity placement [LFZ*23b; LFZ*23a; FOL*21]. These methods aim to minimize area distortion while controlling the number of cones. Cone placement concentrates curvature in isolated points, whereas cut placement distributes it along seams. Approaches range from user-guided [LDB17] to automatic [SC18]. In [SC18], an implicit representation is used, limiting their approach to valence-3 singularities. Further, patches may appear or disappear during the optimization process. These aspects make it unsuitable for approaches where the layout connectivity needs to be preserved.

Parametrization. Parametrization techniques complement layout and cut-based approaches by optimizing embeddings locally within

patches. The main objective is to obtain typically planar embeddings with low distortion. This problem has been studied extensively with methods ranging from linear formulations to non-convex optimization minimizing various distortion measures [SS15]. A comprehensive overview is provided by Sheffer *et al.* [SPR06]. In the context of quad meshing, direction-field-aligned parametrizations are most common [BZK09]. More recently, Corman *et al.* [CC25] introduced a non-linear formulation for rectangular parametrizations. Due to its non-linearity, it is not efficiently applicable in our framework.

Note that our setup is an inversion of the typical parametrization problem: we search for patches corresponding to a prescribed parametrization domain, allowing us to promote a specific shape for the embedded patches, rather than finding a parametrization for a given patch.

3 Layout Embedding

A layout $\mathcal{L}$ is an abstract 2-dimensional cell complex: a combinatorial description of a 2-manifold surface, with no geometric information attached to it. Its connectivity is defined by a simple, connected graph $\mathcal{L} = (\mathcal{N}, \mathcal{A})$, with *nodes* $\mathcal{N}$ and *arcs* $\mathcal{A}$. Additionally, a layout specifies cyclic orderings of arcs around each node n, which induce a set of layout patches $\mathcal{P}$ and their adjacency. The layout $\mathcal{L}$ uniquely determines the topology of the surface represented by $\mathcal{L}$, including its genus g.

As target surfaces $\mathcal{T}$, we consider orientable, closed 2-manifold surfaces of arbitrary genus g. (See Section B, for modifications needed to handle surfaces with boundary.) For a layout to serve as a valid blueprint of a target surface, their topologies must coincide, in particular $g(\mathcal{L}) = g(\mathcal{T})$.

In the continuous setting, an embedding $M : \mathcal{L} \to \mathcal{T}$ is a map that realizes the combinatorial structure of $\mathcal{L}$ on $\mathcal{T}$. Specifically [BSK21]:

- Every node $n \in \mathcal{N}$ is mapped to a distinct point $M(n) \in \mathcal{T}$.
- Every arc $a = (n_1, n_2) \in \mathcal{A}$ is embedded as a continuous path $M(a)$ on $\mathcal{T}$ with $M(n_1)$ and $M(n_2)$ as endpoints.

A layout embedding is *valid* if the following conditions hold:

- The cyclic order of arcs around each node, or equivalently, the patch adjacency, is preserved.
- No two arcs intersect except possibly at shared endpoints.
- The embedded arcs partition the target surface $\mathcal{T}$ into patches each homeomorphic to a disk.

3.1 Problem Statement

We address the following problem: Given a layout $\mathcal{L}$, a target surface $\mathcal{T}$, and an initial *valid* embedding M_0 (see section C), find an embedding M^* that minimizes a given objective function E:

$$M^* = \arg\min_M E(M),$$

subject to the constraint that the embedded path of each arc a remains in the same homotopy class as prescribed by $M_0(a)$, *i.e.*, the embedded arc winds around the surface handles in the same way and with the same number of twists. The objective $E(M)$ integrates over the patches and arcs of $\mathcal{L}$ on $\mathcal{T}$ and is customizable. Our specific formulation is introduced in Section 5.

3.2 Discrete Representation

In practice, we use discrete representations of target surfaces and embeddings. A target surface $\mathcal{T}$ is given by a triangle mesh $(\mathcal{V}, \mathcal{E}, \mathcal{F})$, with vertices $\mathcal{V}$, edges $\mathcal{E}$ and faces $\mathcal{F}$. We denote the 3D positions of vertices of the target surface by $\mathbf{p}_v \in \mathbb{R}^3$ for vertex v.

The layout connectivity $\mathcal{L}$ is specified by its set of nodes $\mathcal{N}$, arcs $\mathcal{A}$ and patches $\mathcal{P}$. Following [NSS*24], each arc is embedded as a piecewise geodesic curve on $\mathcal{T}$. To this end, we sample each arc a, introducing additional nodes that are inserted into $\mathcal{N}$ and thereby subdivide the corresponding arc in $\mathcal{A}$. Nodes can be identified by their valence, with sampled nodes having valence $\deg(n) = 2$ and original corner nodes at patch junctions having valence $\deg(n) \geq 3$. (In Section B, we discuss modifications to handle T-junctions.)

We represent an embedding M of a layout $\mathcal{L}$ into a target surface $\mathcal{T}$ as a tuple $M = (\varphi, \Gamma)$ (see Figure 2), with

$$\varphi : \mathcal{N} \to \mathcal{F} \times \mathbb{R}^2, \qquad \text{(node-to-surface map)}$$

$$\Gamma : \mathcal{A} \to \gamma(a) \subset \mathcal{T}. \qquad \text{(arc-to-curve map)}$$

The node-to-surface map φ assigns to each node of $\mathcal{L}$ a position on the surface $\mathcal{T}$, encoded as barycentric coordinates $\lambda \in \mathbb{R}^2$ in a face $f \in \mathcal{F}$ of the target mesh. (To be precise, barycentric coordinates for points inside a triangular face are $\in \mathbb{R}^3$ and add to one.) The barycentric coordinates of nodes within f constitute the continuous degrees of freedom of our embedding representation.

The arc-to-curve map Γ specifies, for each arc a of $\mathcal{L}$, a geodesic curve $\gamma(a)$ embedded in the target surface $\mathcal{T}$, connecting the endpoints of arcs. The initial curves are provided by a method of choice [MMP87; SC20].

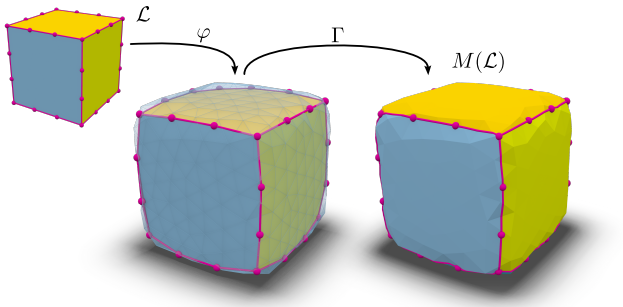


Figure 2: *Embedding representation $M(\mathcal{L})$. For a given layout $\mathcal{L}$, φ maps the magenta layout nodes ($\deg(n) \geq 3$) and sample nodes ($\deg(n) = 2$) to a face on the target surface $\mathcal{T}$. The arc embedding is provided by Γ, making the representation intrinsic.*

4 Differentiable Evaluation of Layout Embeddings

Starting from a given embedding state $M = (\varphi, \Gamma)$, in this section, we discuss how to evaluate the objective function $E(M)$ in a auto-differentiable manner, enabling the computation of gradients w.r.t. the embedding's degrees of freedom, *i.e.*, the node embeddings, which can then be optimized. We collect these degrees of freedom

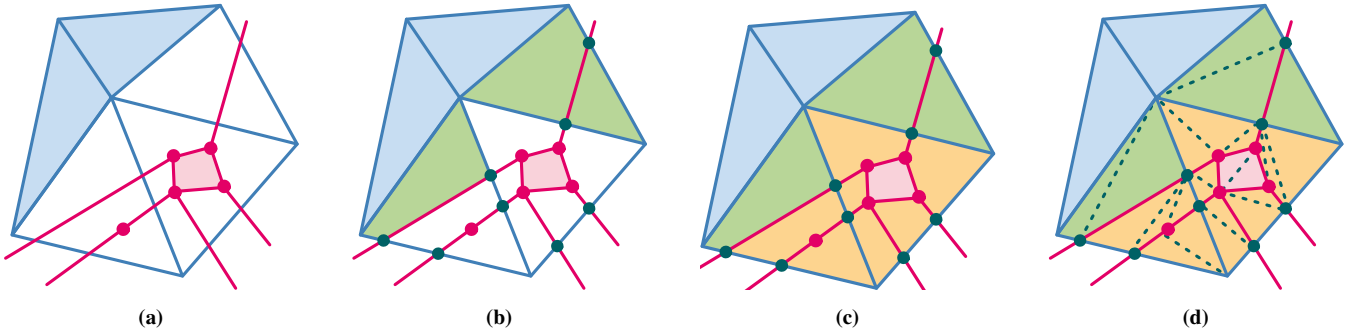


Figure 3: *Obtaining the overlay tessellation. (a) Collection of target faces not touched by layout arcs (blue) and layout faces that lie entirely within a single target face (magenta). (b) Insertion of intersection points (petrol) and collection of polygons bounded by intersection points and target vertices (green). (c) Collecting polygons bounded by layout nodes (orange). (d) Ear clipping of polygonal faces.*

in $\mathbf{x} := \varphi(\mathcal{N})$. The geodesics defining the arcs and patches are determined implicitly by the positions of the embedded nodes. The objective depends not only on the positions of the nodes $\mathcal{N}$ and the geometric embedding of the arcs, but also, crucially, on the geometry of the patches induced by the embedded layout (see Section 5). To ensure differentiability of the resulting patch embeddings derived from M, all operations are designed to allow the computation of derivatives with respect to $\mathbf{x}$.

Overview. The goal is the extraction of the *overlay* $\mathcal{O} = (\mathcal{V}_\mathcal{O}, \mathcal{E}_\mathcal{O}, \mathcal{F}_\mathcal{O})$, *i.e.*, the common triangulation of the target surface $\mathcal{T}$ and the embedded layout induced by $M = (\varphi, \Gamma)$. The overlay enables us to compute the objective function as sum over piecewise linear elements of the overlay mesh. The overlay computation consists of three stages:

(1) **Overlay Geometry:** We extract the overlay geometry by identifying intersection points of arcs a and target mesh edges along the triangle strips traced by their geodesics $\gamma(a)$, see Section 4.1.
(2) **Tessellation:** We adapt the tessellation of $\mathcal{T}$ so that it aligns with the embedded arcs, yielding a triangle mesh overlay $\mathcal{O}$, see Section 4.2.
(3) **Patch Extraction:** We assign each resulting overlay face $f_\mathcal{O}$ to its corresponding layout patch, see Section 4.3.

The overlay provides the combinatorial and geometric backbone for evaluating $E(M)$. During the construction, we carefully track references to the original structures, *e.g.*, which triangles of the overlay constitute the original triangles of the target and which edges correspond to the embedded layout arcs. These references are needed for the evaluation of the objective function.

4.1 Overlay Geometry

The positions $\mathbf{p}_n \in \mathbb{R}^3$ of nodes n on the target surface are obtained directly from the barycentric interpolation of face-vertex positions provided by the node-to-surface map φ. The remaining construction can be carried out independently for each arc $a \in \mathcal{A}$ as follows:

(1) On triangle meshes, geodesic paths are piecewise linear, with segments lying within faces or along edges. From the geodesic $\gamma(a)$, we extract the ordered triangle strip that supports $\gamma(a)$ on $\mathcal{T}$. We embed the strip in $\mathbb{R}^2$ by isometrically unfolding ("puzzling") its triangles. If $\gamma(a)$ passes through a vertex $v \in \mathcal{V}$, the one-ring of faces around v is conceptually split along the path of $\gamma(a)$ giving rise to two half-sectors at v. We select one of the two half-sectors and include it in the strip. For these vertices, the half-sector cannot be embedded isometrically due to nonzero Gaussian curvature at a vertex traversed by $\gamma(a)$. For these faces, we fallback to a greedy embedding approach (see Section A).
(2) To maintain differentiability, intersection parameters $\alpha \in (0,1)$ are computed from arc-edge intersections for this planar embedding of the unfolded strip.
(3) We map the intersection points back to $\mathcal{T}$ and assign them differentiable positions in $\mathbb{R}^3$ based on the intersection edge and parameter α. Together with the node embeddings from φ and the target vertices, they constitute the overlay vertices used in the subsequent tessellation.

4.2 Overlay Tessellation

The overlay tessellation is a layout compatible tesselation of the target surface. To obtain the overlay tessellation, we start by carrying over all target triangles that are not intersected by the embedded layout, and collect all layout patches that lie entirely within a single target triangle, see Figure 3a. For faces intersected by layout arcs, we proceed per target face $f \in \mathcal{F}$:

(1) We begin by inserting the intersection points (Figure 3b). Then, we collect all polygons (green) bounded by the intersection points and vertices of the target mesh.
(2) Further, we collect the polygonal faces bound by intersection points, vertices and layout nodes (orange), see Figure 3c.
(3) Lastly, we triangulate all collected polygons. Since the polygons may be concave, we apply ear clipping to produce the overlay faces $\mathcal{F}_\mathcal{O}$ and overlay edges $\mathcal{E}_\mathcal{O}$ (Figure 3d).

4.3 Patch Extraction

Since we track the correspondence between layout arcs and overlay edges $\mathcal{E}_\mathcal{O}$, we can recover the patches induced by the layout embedding. We mark overlay edges that correspond to layout arcs as patch

boundaries and then perform a flood fill over the overlay faces to obtain the set of layout patches, where each overlay face receives the label of its corresponding layout patch.

5 Objective Functions

In Section 4, we have successfully extracted the overlay tesselation and geometry from a given embedding state M together with a patch labeling per overlay face $f_{\mathcal{O}}$. To make the explanations more accessible, we present the objective functions in terms of overlay mesh elements. It is worth noting that there is a connection to the layout and that the derivatives of the objective functions are computed with respect to the variables $\mathbf{x}$.

While the objective function can be customized for each application and need, we propose an objective based on parametrization quality $E_{\text{dist}}(\mathcal{O})$ (Section 5.1), and a regularization term $E_{\text{align}}(\mathcal{O})$ aligning the patch boundary with a given direction field (Section 5.2). The final objective is the weighted sum of both terms:

$$E(M(\mathcal{L})) = E(\mathcal{O}) = \omega_{\text{dist}} E_{\text{dist}}(\mathcal{O}) + \omega_{\text{align}} E_{\text{align}}(\mathcal{O}).$$

5.1 Parametrization Distortion

In the usual parametrization setting, the goal is to map a surface patch onto a 2D domain. Here, we consider the inverse: starting from a planar domain, we want to determine the best placement of boundaries, which correspond to the embedded layout arcs, for a given boundary shape. For simplicity, we focus on quad layouts, though the approach can be extended to general polygons with only minor adjustments, see Section B.

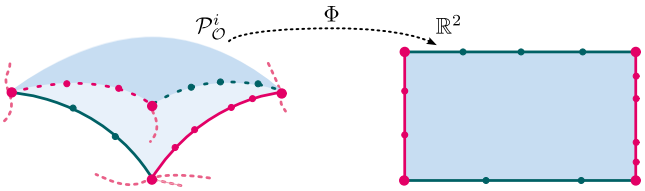


Figure 4: *We compute a planar parametrization for a given patch $\mathcal{P}^i_{\mathcal{O}}$ in the overlay. Note that we prescribe the shape of the parameter domain, i.e., a rectangle, but not its dimensions. The latter are part of the optimization.*

Let $\mathcal{P}^i_{\mathcal{O}}$ denote the i^{th} quadrangular patch in the overlay $\mathcal{O}$ with the boundary originating from the embedding of layout arcs connecting four corner nodes ($\deg(n) \geq 3$), see Figure 4. We heuristically determine the dimensions of the corresponding planar rectangular domain by averaging the embedded lengths of opposite arc sequences. Using these dimensions as boundary conditions, we compute a harmonic parameterization $\Phi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ by solving two Laplace equations:

$$L \begin{bmatrix} \mathbf{u_i} \\ \mathbf{u_b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{u_b^*} \end{bmatrix} \quad \text{and} \quad L \begin{bmatrix} \mathbf{v_i} \\ \mathbf{v_b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{v_b^*} \end{bmatrix},$$

where $\mathbf{u_i}$ and $\mathbf{v_i}$ collect the uv-coordinates of the interior vertices of the patch, and $\mathbf{u_b}, \mathbf{v_b}$ correspond to the vertices along the rectangular boundary subject to boundary position constraints $\mathbf{u_b^*}, \mathbf{v_b^*}$.

The boundary constraints $\mathbf{u_b^*}, \mathbf{v_b^*}$ are obtained by local arc-length parametrization along the boundary arcs. For the Laplacian L, we use the standard cotangent discretization. Note that in practice, the system is reduced by incorporating the boundary conditions into the right-hand side.

The geodesic length of the embedded arcs is computed in a differentiable manner. The same holds for solving the two linear systems to compute Φ. Thus, the dimensions of the rectangle and the parametrization are implicitly part of the optimization.

A wide range of distortion measures based on the map Jacobian $J_\Phi \in \mathbb{R}^{2\times 2}$ are available. Let $\sigma_{\min}$ and $\sigma_{\max}$ be the singular values of Jacobian J_Φ, we use the following point-wise objective for promoting isometry, similar to [APL14]:

$$e_{\text{iso}}(J_\Phi) = \left(\frac{1}{\sigma_{\min}}\right)^2 + (\sigma_{\max})^2.$$

Further, we use a separate term for promoting area preservation:

$$e_{\text{area}}(J_\Phi) = (1 - \sigma_{\min}\sigma_{\max})^2. \tag{1}$$

With Φ defined in a piecewise-linear manner, we have piecewise-constant Jacobians $J_\Phi(f_{\mathcal{O}})$ per overlay face $f_{\mathcal{O}}$. Thus, for the entire patch $\mathcal{P}^i_{\mathcal{O}}$, the distortion energy is given by integrating over the overlay surface:

$$\begin{aligned} E_{\text{dist}}(\mathcal{P}^i_{\mathcal{O}}) &= \int_{\mathcal{P}^i_{\mathcal{O}}} \omega_{\text{iso}}\, e_{\text{iso}}(J_\Phi) + \omega_{\text{area}}\, e_{\text{area}}(J_\Phi)\, dA \\ &= \sum_{f_{\mathcal{O}} \in \mathcal{P}^i_{\mathcal{O}}} |f_{\mathcal{O}}| (\omega_{\text{iso}}\, e_{\text{iso}}(J_\Phi(f_{\mathcal{O}})) + \omega_{\text{area}}\, e_{\text{area}}(J_\Phi(f_{\mathcal{O}}))), \end{aligned} \tag{2}$$

where $|f_{\mathcal{O}}|$ is the area of $f_{\mathcal{O}}$ in $\mathbb{R}^3$.

In parametrization and shape matching, the well-known symmetric Dirichlet energy (SDE) is typically used as the objective. Our setup, however, differs from classical parametrization approaches, making this objective unsuitable. The SDE measures the distortion of both the map Φ and its inverse Φ^{-1}. Hence, part of the integration is performed over the parameter domain. Consequently, this formulation would introduce an incentive for patch shrinkage. Since we aim to preserve patch connectivity, the collapse of patches must be avoided. While the non-symmetric variant of the Dirichlet energy could be considered, we found that having finer control over distortion is more beneficial in our setting. In this sense, e_{area} can be viewed as a regularization term: if Φ must introduce distortion, it should do so in an area-preserving manner.

For the entire overlay $\mathcal{O}$ the distortion energy reads as:

$$E_{\text{dist}}(\mathcal{O}) = \sum_i E_{\text{dist}}(\mathcal{P}^i_{\mathcal{O}}).$$

5.2 Principal Direction Alignment

Our patch-based objective tends to place the layout arcs in regions of high Gaussian curvature. For some applications, it is desirable to control the shape of the boundary further, following the principal curvatures of the surface. To this end, we formulate a direction field alignment term.

We compute a smooth face-based 4-rosy direction field on $\mathcal{T}$ based on the principal curvature directions. This can be done with a

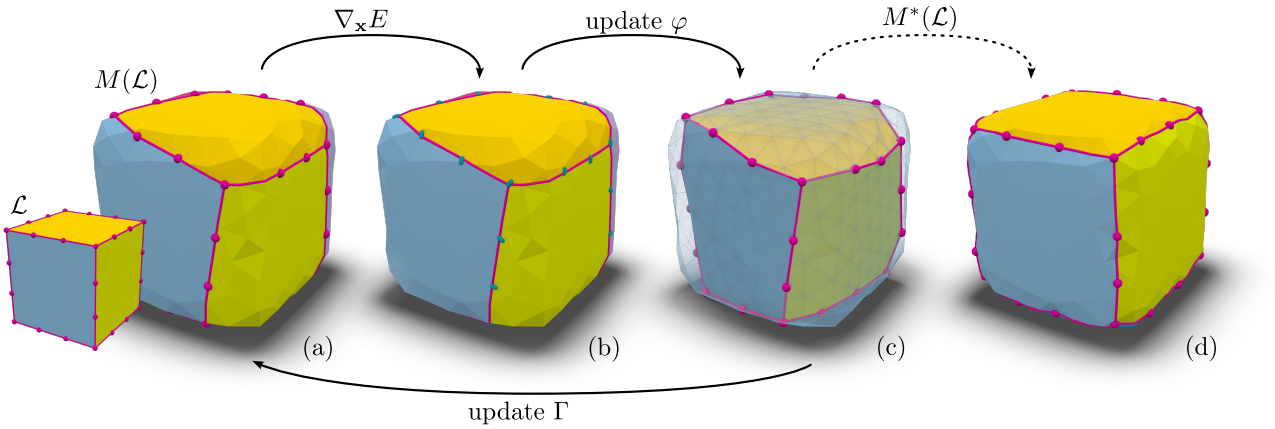


Figure 5: *Steps of our embedding optimization. (a) For a given embedding $M(\mathcal{L})$, we evaluate the gradient $\nabla_{\mathbf{x}} E$. (b) Using the gradient information, we compute update directions* $\mathbf{u}$ *(petrol directions). (c) Using the Levi-Civita connection of the surface and the direction* $\mathbf{u}$*, we update* φ*. With the updated map* φ*, we recompute the geodesic connections* Γ *from scratch and the process is repeated. After convergence (d), we obtain the final embedding $M^*(\mathcal{L})$, ready to be used in downstream applications.*

method of choice [KCPS13; PCS24]. Following the representation in [KCPS13], the field orientation is represented by a single complex number $z^*(f) \in \mathbb{C}$ per target face $f \in \mathcal{F}$. The field orientation z^* is found by raising any of the four field directions to the 4^{th} power, effectively making directions differing by 90° indistinguishable from each other.

Using this smooth direction field, the objective can be formulated per overlay edge $e_{\mathcal{O}} \in \mathcal{E}_{\mathcal{O}}(\mathcal{A})$ originating from a layout arc. First, we look up the target face f, in which $e_{\mathcal{O}}$ is embedded. The direction of the edge is computed w.r.t. the reference edge and raised to the 4^{th} power, yielding $z \in \mathbb{C}$. Now, z is compared to the target orientation $z^*(f)$, so the alignment term is given by:

$$E_{\text{align}}(\mathcal{A}) = \frac{1}{|\mathcal{E}_{\mathcal{O}}(\mathcal{A})|} \int_{\mathcal{A}} e_{\text{align}}(s)\, ds \tag{3}$$

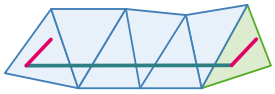
$$= \frac{1}{|\mathcal{E}_{\mathcal{O}}(\mathcal{A})|} \sum_{e \in \mathcal{E}_{\mathcal{O}}(\mathcal{A})} |e| \left(z - z^*(f)\right)^2, \tag{4}$$

where $|e|$ is the length of an edge in the overlay mesh in $\mathbb{R}^3$ and $\frac{1}{|\mathcal{E}_{\mathcal{O}}(\mathcal{A})|}$ normalizes by the total length of embedded arcs.

6 Optimization Strategy

With the construction in Section 4, we compute derivatives of the objective function in Section 5 for optimization w.r.t. the degrees of freedom $\mathbf{x} = \varphi(\mathcal{N})$ associated with the nodes $\mathcal{N}$ of the layout. In this section, we discuss the optimization procedure depicted in Figure 5.

Momentum Estimation. Our objective function is non-convex and not continuous. To mitigate the effects of derivative discontinuities, we use a momentum-based gradient-descent procedure. After computing $\nabla_{\mathbf{x}}^{(i)} E$ at iteration i via backpropagation using *LibTorch* [AYH*24], we compute the update directions similarly to VectorAdam [LSJ22]. In the original formulation, the gradient directions are defined in ambient space. Since our variables $\mathbf{x}$ are restricted to the tangent space of the surface, special care has to be taken when updating the first and second moments. They must be expressed in the same tangent space as the gradient, which can be achieved using parallel transport (PT) via the surface's Levi-Civita connection [dGDT16]. More precisely, whenever we move from one triangle to another (when applying an update step), we unfold the triangles (see inset figure). In the unfolded setting, the transport of vectors, corresponding to the optimizer's state (magenta), is a simple parallel transport (petrol).



Update of φ and Γ. We update the vertex-to-surface map φ as follows:

$$\mathbf{x}^{(i+1)} \leftarrow \text{PT}\left(\mathbf{x}^{(i)},\, s \cdot \mathbf{u}\left(\nabla_{\mathbf{x}}^{(i)} E, i\right)\right),$$

where s is a user-defined step-size. The update direction $\mathbf{u}$ is computed by our adapted VectorAdam strategy, which is also expressed in tangent space of the target surface $\mathcal{T}$. Since the nodes $\mathbf{x}^{(i)}$ are restricted to the surface, we have to trace the update vector $\mathbf{u}$ along the target surface via the same parallel transport operator used above to obtain the new positions $\mathbf{x}^{(i+1)}$ on the target surface. After updating the map φ, the piecewise geodesic embedding of the layout arcs $\mathcal{A}$ is recomputed.

Arc Resampling. After several optimization iterations the samples are unevenly distributed along arcs. In regions with fewer sample points, the flexibility of the arcs is reduced. In regions with denser samples, the probability for self-intersections after an update is higher. In order to maintain flexibility and a larger step-size, we uniformly re-sample the arcs after a certain number of iterations. Since this introduces new optimization variables, we have to re-initialize

the optimizer's state for all degrees of freedom. To mitigate the effect of noisy gradient directions, we scale down s after re-initialization and linearly increase it to the original step-size over several iterations, effectively implementing a warm-up strategy. To avoid costly backtracking in the case of intersecting embedded arcs, we remove sample nodes whose geodesic distance to an adjacent node falls below a threshold. To preserve patch neighborhood relations, layout corner nodes with $\deg(n) \geq 3$ are not merged.

Patch Validity. To avoid the collapse of a patch, we enforced a minimum side length for the sides of the parameter domain. While the described strategies (resampling, collapsing and minimum side length) do not guarantee that arcs remain non-intersecting and that patches don't collapse after an update, we rarely observed the need for backtracking in practice when using a sufficiently small step size s.

7 Evaluation and Applications

In the following, we evaluate the proposed method for layout embedding optimization. Figure 6 presents a collection of results. We begin by analyzing robustness with respect to the initial embedding and noise (Section 7.1). Further, we compare our objective function to an alternative choice and highlight the effect of the direction-field alignment term (Section 7.2). In Section 7.3, we demonstrate that optimized layouts can be transferred across different mesh resolutions. We show that our method preserves the combinatorial structure of layouts and compare it to the approach of Sharp and Crane [SC18] (Section 7.4). Further, we present applications where the improved embeddings are beneficial, *i .e.*, q uad m eshing and inter-surface mapping, and compare our method to alternative approaches (Sections 7.5 and 7.6). Lastly in Section 7.7, we report on the performance of the algorithm .

Evaluation Setup. For all experiments, we isotropically retriangulate the input surfaces and normalize their total area to 1. The optimization is primarily driven by E_{dist} with a fixed weight $\omega_{\textbf{dist}} = 1.0$. The individual weights were set to $\omega_{\textbf{area}} = 0.5$ and $\omega_{\textbf{iso}} = 0.5$, which yielded consistent results. The alignment term is mostly used for regularization, with weights $0.0 \leq w_{\text{align}} \leq 0.5$. Most commonly, we used $w_{\text{align}} = 0.1$.

For Adam optimization, we use $\beta_1 = 0.9$, $\beta_2 = 0.9$, and $\epsilon = 10^{-8}$, with a maximum step size of $s = 1.5 \times 10^{-3}$ combined with a warmup strategy.

Resampling is performed with a target edge length dependent on the input surface, typically around 0.03, but each arc is subdivided at least once. Resampling is initially applied after 5 iterations and then the interval is always increased by 15 iterations after resampling. This schedule allows us to preserve the optimizer's internal state for longer, when updates become smaller and resampling is less necessary.

The initial embeddings are either created manually or derived from a given quad mesh. Quad meshes are computed using the method of Lyon et al. [LCK21b]. Note that our approach can be integrated into the pipeline of any quad-meshing algorithm, prior to the extraction of the quad mesh.

7.1 Robustness

We evaluate the robustness of our method w.r.t. both initialization and noise.

Initialization. To demonstrate flexibility, we selected multiple layouts for the same target surface. In all cases, our optimization converges to the expected embedding, see Figure 7. The top row shows the optimized embeddings, while the bottom row depicts the corresponding initial embeddings. The objective preserves patches: a single patch per cube face would suffice, yet none of the 3×3 patches collapse. It also handles asymmetric setups and finds a good solution even when multiple patches start from an initial embedding far from the optimum.

Noise. We demonstrate that our objective function supports reliable optimization even in the presence of noise, compared to [CK14b], which fails in this setting because it relies on principal curvature information, which is not robust to noise. For this experiment, we remesh a target surface isotropically and perturb it by displacing vertices along their normals. The displacement magnitudes are sampled from a uniform distribution and scaled relative to the average edge length. As shown in Figure 8, our method produces reasonable embeddings across a range of noise levels. For higher levels of noise, however, the construction of the initial solution may yield invalid embeddings, preventing successful optimization.

7.2 Objective Function

We perform an ablation of our main distortion minimizing energy term by replacing it with the Hencky energy used in [SC18], followed by a discussion of how the direction field alignment term affects the solution.

Hencky Energy. Instead of explicitly parametrizing the surface to measure distortion, an elegant alternative, used in [SC18], is to solve the Yamabe equation, which provides the scale factor u for the metric resulting from a conformal parametrization. Since a conformal parametrization preserves angles, the scale factor u can be used as measure for distortion. Compared to our approach, this requires solving only a single Poisson equation, whereas our method involves two and a singular value decomposition of the Jacobian for each triangle in $\mathcal{O}$. We implemented the Hencky energy

$$E_H = \int_\Omega u^2 \, \mathrm{d}s,$$

as used in [SC18], within our framework and compared it to our objective in Figure 9. As already pointed out in [SC18], minimizing E_H is ill-posed: the distortion can always be further reduced by introducing longer cuts. While our framework would allow the preservation of the patch connectivity, the patch shapes hinder the use of the embedding in downstream applications.

Direction Field Alignment. The primary objective of our formulation is the distortion energy E_{dist}. While the optimization reliably minimizes E_{dist} (Figure 10 (a)), the resulting embeddings may not always match with human intuition. To address this, the additional

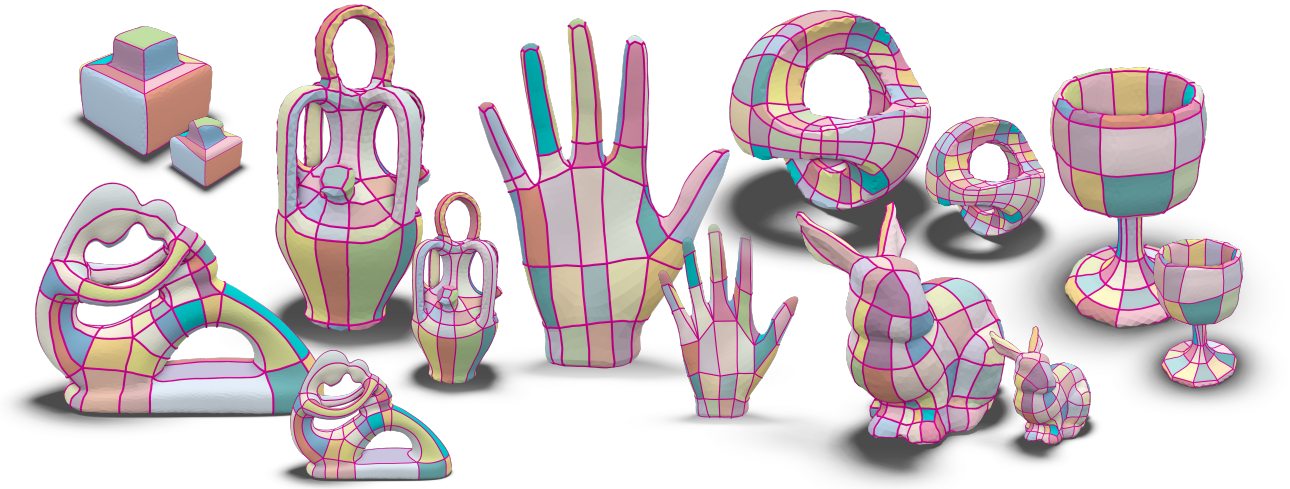


Figure 6: *Collection of layout embeddings optimized with our method, shown alongside their initial embeddings (small inset figures). The results demonstrate our method's ability to reposition layout nodes to achieve higher-quality embeddings while preserving the input layout's connectivity.*

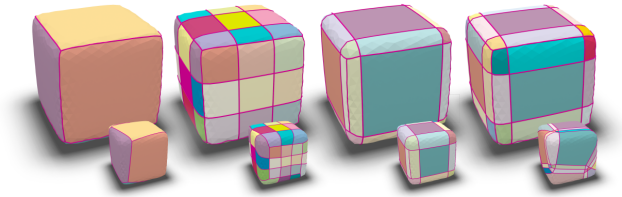
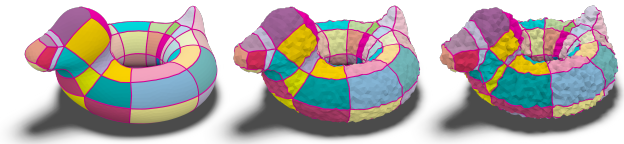


Figure 7: *Different layouts $\mathcal{L}$ for the same target surface $\mathcal{T}$. Top: optimized embeddings. Bottom: initial embeddings. Note that even with superfluous patches or unfavorable initialization, our optimization is able to converge to the expected embedding.*

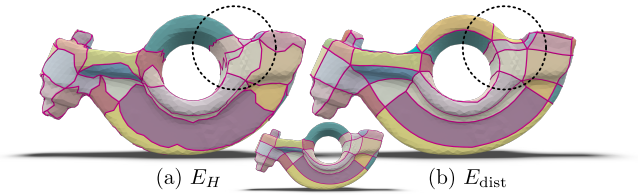


(a) 0% Noise (b) 1% Noise (c) 2% Noise

Figure 8: *We show the robustness of our optimization under different noise levels.*

direction field alignment term E_{align} operates as a regularizer, biasing the optimization toward embeddings that also respect the direction field (Figure 10 (b-c)).

The quality of the direction field, in particular the locations of its singular points, affects the final embedding. Some patch corners in the layout are attracted to these singularities (see Figure 11). When singularities are misplaced, for example due to asymmetry, the resulting layout adapts to this undesirable placement.



(a) E_H (b) E_{dist}

Figure 9: *Comparison of Hencky energy E_H and our distortion objective E_{dist}. E_H tends to increase the length of the embedded arcs, while E_{dist} keeps the patches rectangular.*

7.3 Multi-resolution

To transfer a layout between low- and high-resolution models, it suffices to transport the map φ and recompute the geodesic connections. This is straightforward, as our representation is independent of mesh resolution. An example is shown in Figure 12.

7.4 Preservation of Combinatorial Structure

The explicit representation has several advantages over implicit representations, *e.g.*, used in [SC18], hereafter referred to as VSC. First, we can represent singularities of arbitrary valence, whereas in VSC, they are restricted to valence 3. Already when converting our representation to the implicit one used in VSC, the original patch connectivity is lost (see small inset figures in Figure 13), both because VSC can only handle valence 3 and because their representation is mesh-resolution dependent.

While in theory the flow optimized in VSC does not alter patch connectivity, in practice this cannot be guaranteed (see Figure 13). Consequently, VSC is unsuitable for applications such as quad meshing.

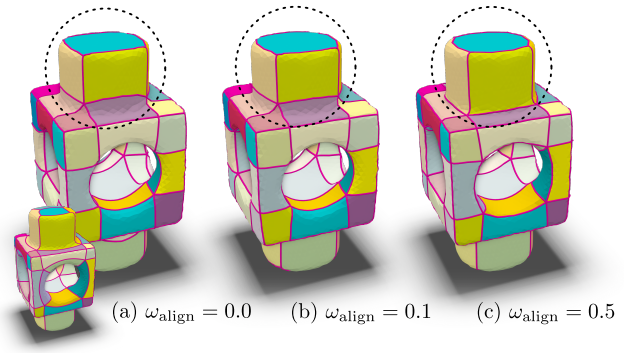


Figure 10: *Starting from the same initial solution (small inset figure), we optimized the embedding with different weights of the alignment term E_{align}. While the patches in (a) have lower distortion under our metric, the patches in (b) and (c) better align with principal curvature.*

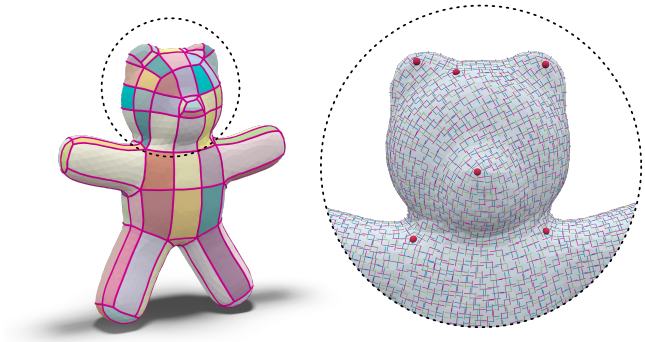


Figure 11: *Using our alignment term E_{align}, an asymmetric direction field (right, with singularities in red) implies an asymmetric layout embedding (left).*

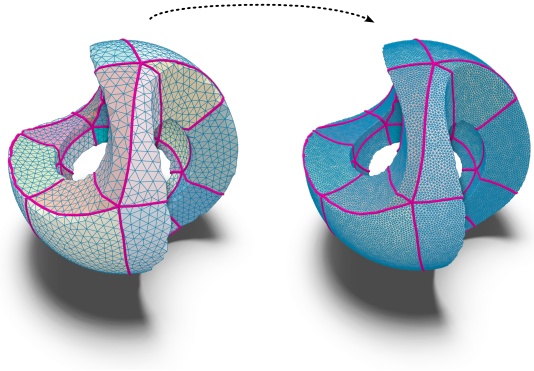


Figure 12: *A result can trivially be transported across different target mesh resolutions, since our representation is independent of the mesh resolution.*

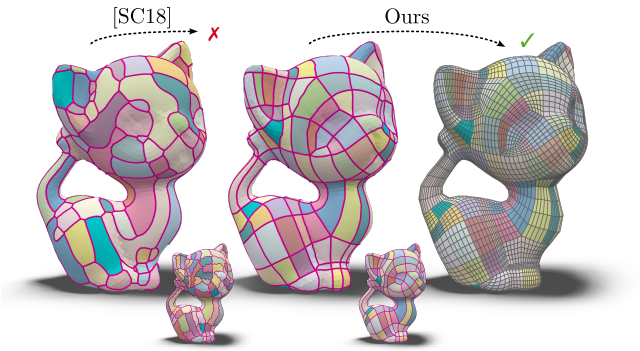


Figure 13: *Comparison of [SC18] (VSC) and our method. By preserving patch connectivity, our optimized layout embeddings can be used in downstream applications. Note that VSC is restricted to valence 3 layout nodes, which requires a slightly different initialization to our method.*

7.5 Quad Meshing

Typical parametrization-based approaches for quad meshing operate in multiple stages. Most state-of-the-art methods begin by computing a direction field, which is then used to produce a parametrization. At some point during their process, all approaches must settle on a layout, induced by the uv-isolines connecting singularities. Typically, once singularities are detected in an early stage, they remain fixed throughout the pipeline.

We show that introducing our embedding optimization as an additional step after the extraction of the quad layout improves the quality of the resulting quad mesh. In our experiments, allowing singularities to move freely yields higher mesh quality than using Laplacian smoothing as a postprocessing step to relocate the singularities, as done in [LCK21b]. As demonstrated in Figure 14, the resulting quad meshes exhibit higher quality, even though the layout connectivity is identical and originates from [LCK21b].

For evaluation, we measured both the minimum scaled Jacobian J and the maximum inner angle per quad as quality indicators. Our quad meshes were generated using the following pipeline:

(1) Embed the provided quad layout into the target surface and perform our optimization.
(2) Select a number of subdivisions per dual loop.
(3) Compute a harmonic parametrization of each patch with a rectangular boundary, effectively producing an integer-grid map. The dimensions are prescribed by the dual loop subdivisions.
(4) Extract the quad mesh from the integer-grid map by mapping vertices at integer locations back to the input surface.

7.6 Inter-surface Maps

The computation of correspondences between surfaces is a well-studied problem with a wide range of approaches. Most methods struggle with highly non-isometric relations between shapes, as they typically rely on intrinsic regularization terms, *e.g.*, geodesic length preservation or map distortion. For our comparison, we selected the distortion-driven approach of Schmidt *et al.* [SPK23], which seeks correspondences that minimize the overall distortion introduced by the map in a least-squares sense. In addition to the inter-surface map, this method also produces a compatible triangulation of the input surfaces. To visualize the correspondence, we cluster normals

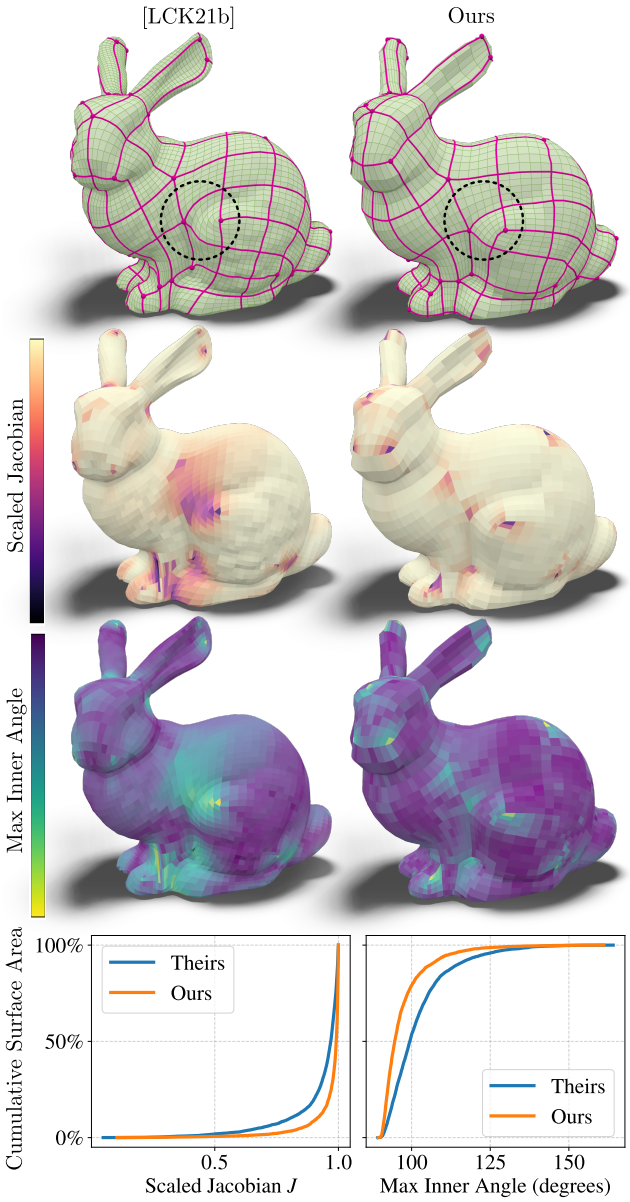


Figure 14: *Comparison of a quad mesh generated from the original layout embedding of [LCK21b] with one produced using our optimized embedding. Note that their result already includes a post-processing smoothing step.*

on one surface and transport the clustering to the others via the inter-surface map, see Figure 15 (left). Since the triangulations are compatible, there is a one-to-one correspondence between faces, making this transport trivial.

A layout can be viewed as a topological template without fixed geometry, unlike templates used in typical template-based shape correspondence approaches. We compute maps from each layout patch to the target surfaces. Since patches may differ in size between

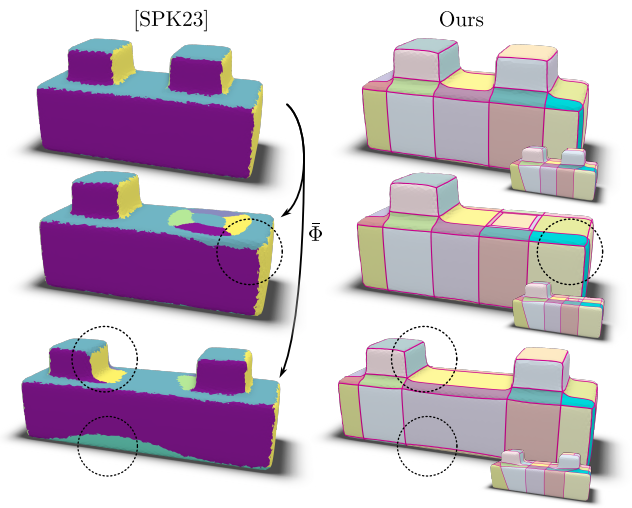


Figure 15: *Comparison of correspondences generated by [SPK23] and by our approach. Starting from initial embeddings of a common layout $\mathcal{L}$ (small inset figures), our method produces a map that adapts to the non-isometric deformation between the shapes, whereas the map of [SPK23] distributes distortion in a least-squares sense.*

the two targets, our method can introduce distortion where necessary (see Figure 15) and overall respect features such as creases.

The correspondence is prescribed only at a coarser, global level. A true point-to-point map can then be obtained by applying any distortion-based method within each patch. Figure 16 shows results on a second pair of input surfaces. Effectively, our method produces a co-segmentation, preparing the surfaces for various co-processing tasks such as compatible remeshing. Overall, these results demonstrate the strength of our approach in producing meaningful correspondences even under challenging non-isometric conditions.

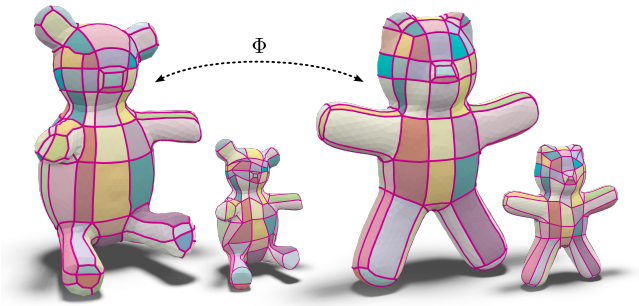


Figure 16: *Co-segmentation of two teddy models (initial embeddings shown as small inset figures). The relation between the two models is non-isometric. Our layout embeddings respect this relation, as patches are embedded with different dimensions on the two surfaces. (Note: the embedding of the right teddy was obtained without direction alignment; see Section 7.2.)*

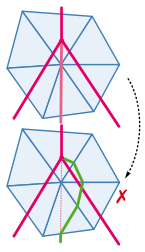
7.7 Runtime Analysis

The proposed layout embedding optimization algorithm was implemented in C++. All experiments ran on a system with an AMD Ryzen 9 5900X CPU (12 cores) and 62 GB of RAM, running Debian GNU/Linux 13. Table 1 summarizes the measured runtimes and analyzes the distribution of computational costs across different parts of the algorithm. From the table, we observe that the runtime per iteration, as expected, depends on the number of faces of the target surface and the number of patches in the layout. To better understand the computational effort per iteration, we analyze the portion of time spent in the main components of the algorithm: embedding the layout to obtain the overlay mesh, evaluating the objective function, and computing gradients via backpropagation. We find that the runtime is dominated by the backpropagation step. For meshes with a larger number of faces and more complex layout connectivity, the embedding step also contributes a more significant portion of the total runtime.

8 Limitations and Future Work

In the previous sections, we motivated the need for an embedding optimization method that preserves a prescribed layout connectivity and showed that our method satisfies this requirement. Further, we showed the benefits of our optimized embeddings for down-stream applications.

Nevertheless, the implementation of our proposed method could be improved in the future. Currently, a large portion of compute is spent in the backpropagation step. Replacing the auto-differentiation bears potential for performance improvements, making our method more efficient. Moreover, the method suffers from limited numerical robustness. One source are badly shaped triangles near the embedded layout arcs in the overlay mesh. Lastly, a backtracking procedure would increase the robustness of the implementation, allowing the algorithm to recover from invalid embeddings encountered after an update step.



While our construction for differentiable geodesics (Section A) works well in practice, it happens in rare cases that, after our modification step, the cyclic ordering around the layout nodes is not preserved (see inset figure). In this case, the embedding is invalid. It is worth noting that a modification that preserves said ordering always exists. It would therefore be interesting to investigate different strategies or to find a different formulation that connects the nodes. One possibility is to compute geodesics under a constant curvature metric as in [SCBK20]. However, the constant curvature connects nodes unintuitively. To circumvent this, a denser sampling would be needed.

As a general direction for future work, it would also be worthwhile to explore objectives that are specifically tailored to an application. For instance, one could investigate how to incorporate semantic labels for shape correspondences.

Acknowledgments

This work was partially funded by the German Research Foundation within the Gottfried-Wilhelm-Leibniz program and by the Excellence Strategy of the Federal Government and the States of Germany. Open Access was funded by the RWTH-publication funds. Further, we thank Philip Trettner for maintaining the *glow* and *polymesh* libraries.

Appendix

A Differentiable Geodesic Paths

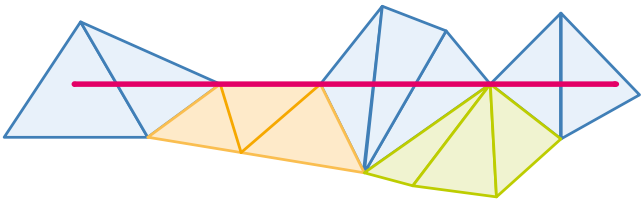


Figure 17: *Unfolding of triangle strip supporting the geodesic (magenta) of an arc* $\gamma(a)$.

Our method depends on finding differentiable intersections of the faces of $\mathcal{T}$ with layout arcs. We start by unfolding the ordered triangle strip that supports $\gamma(a)$ on $\mathcal{T}$, obtaining the blue triangles in Figure 17. If a geodesic $\gamma(a)$ crosses vertices in $\mathcal{T}$, we need to ensure that the intersection parameter α is in $(0,1)$ and that there is a unique edge associated with the intersection. To this end, we apply the following construction: If only a single vertex is crossed, we greedily add one triangle fan to the triangle strip (green faces). The embedding cannot be isometric, therefore, we embed them considering the available angle. If multiple vertices are crossed, we greedily select one of the triangle strips containing these vertices (orange faces) and embed them in the available space. After the embedding, we shift the vertices that were the original intersection points slightly upwards. Now, the geodesic (magenta line) intersects the unfolded triangle strip, yielding intersection parameters $\alpha \in (0,1)$.

B General Layout Patches

The proposed method can be readily extended to non-quadrangular patches, layouts with T-junctions, and layouts embedded in target surfaces with boundaries. In the following, we describe the necessary adaptations required to handle these cases.

Non-quadrangular patches. For handling non-quadrangular patches, the main difference lies in defining the appropriate boundary conditions for the harmonic parametrization, with the sole requirement that the resulting boundary must be convex in the parameter domain. Under this condition, the resulting harmonic parametrization is guaranteed, at least theoretically, to be bijective, *i.e.*, free of flipped triangles. The specific boundary shape may be chosen depending on the application. For triangular patches, one option is to use the embedded edge lengths as side lengths. If these lengths violate the triangle inequality, a robust fallback is to use an equilateral triangle whose edge length is given by the average embedded edge length.

Quadrangular patches with T-junctions. To support T-junctions, it is necessary to identify the layout nodes forming the junction, as

Model	#Faces	#Patches	Total (min)	(s)/Iter.	Embed. (%)	Eval. (%)	BP (%)
SHREC07-177	4,176	114	36	12.1	15.9	5.3	72.1
SHREC07-169	4,642	114	52	12.3	18.6	5.2	69.8
bunny	6,000	148	94	16.0	20.8	4.2	68.2
SHREC07-35	6,522	113	31	14.3	14.9	5.1	73.6
hand (remeshed)	1,480	78	32	4.9	12.4	5.7	74.8
hand	6,400	78	90	14.0	18.6	4.8	70.3
bob	10,642	87	76	22.8	24.2	4.1	65.7
kitten	11,320	198	157	33.3	31.5	3.4	59.2
spot	17,018	206	173	58.8	44.8	2.8	47.2
Average					22.4	5.1	66.7

Table 1: *Performance summary of the algorithm, reporting face and patch counts, total runtime, per-iteration time, and percentage of overall runtime for selected components. Percentages may not sum to* 100% *since some minor parts of the algorithm are excluded. The percentages for Embedding, Evaluation, and Backpropagation indicate the share of total runtime spent computing the overlay mesh in a differentiable way, evaluating the objective function, and computing gradients, respectively. The reported averages show that runtime is dominated by the backpropagation step.*

well as the half-edges along the T-base and T-bar. This information is typically provided by methods that explicitly construct T-junction layouts. T-nodes are then treated as patch corners in the two patches adjacent to the T-base, while in the patch over the T-bar, they are treated as arc samples. During resampling, T-junction nodes are handled like regular layout nodes and must be preserved.

Target surface with boundary. When the target surface has a boundary, layout nodes lying on the boundary require special handling. If the layout is intended to cover the entire surface, these nodes and the arcs connecting two boundary nodes must be constrained to the boundary of the target surface. At the implementation level, this implies that boundary nodes are embedded in boundary edges rather than interior faces. During the update procedure, boundary layout nodes are restricted to move along the boundary, traversing successive boundary edges according to the length of the update direction.

C Input Layout Validity

Given a layout $\mathcal{L}$ and an initial embedding of the layout nodes, the geodesic arc embeddings may intersect or the cyclic order around a node may be violated. In such a case, the layout embedding M is *invalid* and the algorithm cannot extract valid patch embeddings. If an invalid embedding is encountered, a procedure can be used to modify the layout in such a way that the resulting embedding will be valid. To this end, any embedding procedure can be used that guarantees a valid embedding, *e.g.*, the layout embedding procedure by [BSK21]. This embedding procedure will produce an embedding for the layout arcs, which can now be sampled such that a piecewise geodesic embedding will result in a valid embedding.

Acknowledgment

Open Access funding enabled and organized by Projekt DEAL.

References

[APL14] AIGERMAN, NOAM, PORANNE, ROI, and LIPMAN, YARON. "Lifted bijections for low distortion surface mappings". *ACM Transactions on Graphics* 33.4 (2014), 69:1–69:12 5.

[AYH*24] ANSEL, JASON, YANG, EDWARD, HE, HORACE, et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ACM, 2024, 929–947 6.

[BSK21] BORN, JANIS, SCHMIDT, PATRICK, and KOBBELT, LEIF. "Layout Embedding via Combinatorial Optimization". *Computer Graphics Forum* 40.2 (2021), 277–290 2, 3, 12.

[BZK09] BOMMES, DAVID, ZIMMER, HENRIK, and KOBBELT, LEIF. "Mixed-integer quadrangulation". *ACM Transactions on Graphics* 28.3 (2009), 77 3.

[Cam17] CAMPEN, MARCEL. "Partitioning Surfaces Into Quadrilateral Patches: A Survey". *Computer Graphics Forum* 36.8 (2017), 567–588 2.

[CC25] CORMAN, ETIENNE and CRANE, KEENAN. "Rectangular Surface Parameterization". *ACM Transactions on Graphics* 44.4 (2025), 111:1–111:21 3.

[CDH*24] COUDERT-OSMONT, YOANN, DESOBRY, DAVID, HEISTERMANN, MARTIN, et al. "Quad Mesh Quantization Without a T-Mesh". *Computer Graphics Forum* 43.1 (2024) 2.

[CK14a] CAMPEN, MARCEL and KOBBELT, LEIF. "Dual strip weaving: interactive design of quad layouts using elastica strips". *ACM Transactions on Graphics* 33.6 (2014), 183:1–183:10 2.

[CK14b] CAMPEN, MARCEL and KOBBELT, LEIF. "Quad Layout Embedding via Aligned Parameterization". *Computer Graphics Forum* 33.8 (2014), 69–81 2, 7.

[dGDT16] De GOES, FERNANDO, DESBRUN, MATHIEU, and TONG, YIYING. "Vector field processing on triangle meshes". *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '16*. ACM, 2016, 27:1–27:49 6.

[FOL*21] FANG, QING, OUYANG, WENQING, LI, MO, et al. "Computing sparse cones with bounded distortion for conformal parameterizations". *ACM Transactions on Graphics* 40.6 (2021), 262:1–262:9 2.

[KCPS13] KNÖPPEL, FELIX, CRANE, KEENAN, PINKALL, ULRICH, and SCHRÖDER, PETER. "Globally optimal direction fields". *ACM Transactions on Graphics* 32.4 (2013), 59:1–59:10 6.

[LCK21a] LYON, MAX, CAMPEN, MARCEL, and KOBBELT, LEIF. "Quad Layouts via Constrained T-Mesh Quantization". *Computer Graphics Forum* 40.2 (2021), 305–314 2.

[LCK21b] LYON, MAX, CAMPEN, MARCEL, and KOBBELT, LEIF. "Simpler Quad Layouts using Relaxed Singularities". *Computer Graphics Forum* 40.5 (2021), 169–179 2, 7, 9, 10.

[LDB17] LUCQUIN, VICTOR, DEGUY, SÉBASTIEN, and BOUBEKEUR, TAMY. "SeamCut: interactive mesh segmentation for parameterization". *SIGGRAPH Asia Technical Briefs*. Ed. by GUTIERREZ, DIEGO and HUANG, HUI. ACM, 2017, 25:1–25:4 2.

[LFZ*23a] LI, MO, FANG, QING, ZHANG, ZHENG, et al. "Efficient Cone Singularity Construction for Conformal Parameterizations". *ACM Transactions on Graphics* 42.6 (2023), 235:1–235:13 2.

[LFZ*23b] LI, MO, FANG, QING, ZHANG, ZHENG, et al. "Efficient cone singularity construction for conformal parameterizations". *ACM Transactions on Graphics (TOG)* 42.6 (2023), 1–13 2.

[LNTC24] LI, YUE, NUMEROW, LOGAN, THOMASZEWSKI, BERNHARD, and COROS, STELIAN. "Differentiable Geodesic Distance for Intrinsic Minimization on Triangle Meshes". *ACM Transactions on Graphics* 43.4 (2024), 91:1–91:14 2.

[LSJ22] LING, SELENA, SHARP, NICHOLAS, and JACOBSON, ALEC. "VectorAdam for Rotation Equivariant Geometry Optimization". *Advances in Neural Information Processing Systems*. Vol. 35. 2022 6.

[MMP87] MITCHELL, JOSEPH S. B., MOUNT, DAVID M., and PAPADIMITRIOU, CHRISTOS H. "The Discrete Geodesic Problem". *SIAM Journal on Computing* 16.4 (1987), 647–668 3.

[NSS*24] NOMA, YUTA, SELLÁN, SILVIA, SHARP, NICHOLAS, et al. "Surface-Filling Curve Flows via Implicit Medial Axes". *ACM Transactions on Graphics* 43.4 (2024), 147:1–147:12 2, 3.

[PBS22] PANDEY, KARRAN, BÆRENTZEN, JAKOB ANDREAS, and SINGH, KARAN. "Face Extrusion Quad Meshes". *SIGGRAPH: Special Interest Group on Computer Graphics and Interactive Techniques Conference*. Ed. by NANDIGJAV, MUNKHTSETSEG, MITRA, NILOY J., and HERTZMANN, AARON. ACM, 2022, 10:1–10:9 2.

[PCS24] PALMER, DAVID R., CHERN, ALBERT, and SOLOMON, JUSTIN. "Lifting Directional Fields to Minimal Sections". *ACM Transactions on Graphics* 43.4 (2024), 60:1–60:20 6.

[PNA*21] PIETRONI, NICO, NUVOLI, STEFANO, ALDERIGHI, THOMAS, et al. "Reliable feature-line driven quad-remeshing". *ACM Transactions on Graphics* 40.4 (2021), 155:1–155:17 2.

[PPM*16] PIETRONI, NICO, PUPPO, ENRICO, MARCIAS, GIORGIO, et al. "Tracing Field-Coherent Quad Layouts". *Computer Graphics Forum* 35.7 (2016), 485–496 2.

[PSS01] PRAUN, EMIL, SWELDENS, WIM, and SCHRÖDER, PETER. "Consistent mesh parameterizations". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 2001, 179–184 2.

[SAPH04] SCHREINER, JOHN, ASIRVATHAM, ARUL, PRAUN, EMIL, and HOPPE, HUGUES. "Inter-surface mapping". *ACM Transactions on Graphics* 23.3 (2004), 870–877 2.

[SC18] SHARP, NICHOLAS and CRANE, KEENAN. "Variational surface cutting". *ACM Transactions on Graphics* 37.4 (2018), 156 2, 7–9.

[SC20] SHARP, NICHOLAS and CRANE, KEENAN. "You can find geodesic paths in triangle meshes by just flipping edges". *ACM Transactions on Graphics* 39.6 (2020), 1–15 3.

[SCBK20] SCHMIDT, PATRICK, CAMPEN, MARCEL, BORN, JANIS, and KOBBELT, LEIF. "Inter-surface maps via constant-curvature metrics". *ACM Transactions on Graphics* 39.4 (2020), 119 2, 11.

[SPK23] SCHMIDT, PATRICK, PIEPER, DÖRTE, and KOBBELT, LEIF. "Surface Maps via Adaptive Triangulations". *Computer Graphics Forum* 42.2 (2023), 103–117 9, 10.

[SPR06] SHEFFER, ALLA, PRAUN, EMIL, and ROSE, KENNETH. "Mesh Parameterization Methods and Their Applications". *Foundations and Trends in Computer Graphics and Vision* 2.2 (2006) 3.

[SS15] SMITH, JASON and SCHAEFER, SCOTT. "Bijective parameterization with free boundaries". *ACM Transactions on Graphics* 34.4 (2015), 70:1–70:9 3.

[TPP*11] TARINI, MARCO, PUPPO, ENRICO, PANOZZO, DANIELE, et al. "Simple quad domains for field aligned mesh parametrization". *ACM Transactions on Graphics* 30.6 (2011), 142 2.

[TPSS13] TAKAYAMA, KENSHI, PANOZZO, DANIELE, SORKINE-HORNUNG, ALEXANDER, and SORKINE-HORNUNG, OLGA. "Sketch-based generation and editing of quad meshes". *ACM Transactions on Graphics* 32.4 (2013), 97:1–97:8 2.

[ULP*15] USAI, FRANCESCO, LIVESU, MARCO, PUPPO, ENRICO, et al. "Extraction of the Quad Layout of a Triangle Mesh Guided by Its Curve Skeleton". *ACM Transactions on Graphics* 35.1 (2015), 6:1–6:13 2.