Free-form Surface Approximation Using Rotational Patches

YUANPENG LIU, RMIT University, Melbourne, Australia
YI MIN XIE, Hohai University, Changzhou, China
TING-UEI LEE, RMIT University, Melbourne, Australia
ZIQI WANG, The Hong Kong University of Science and Technology, Hong Kong, China
NICO PIETRONI, University of Technology Sydney, Sydney, Australia

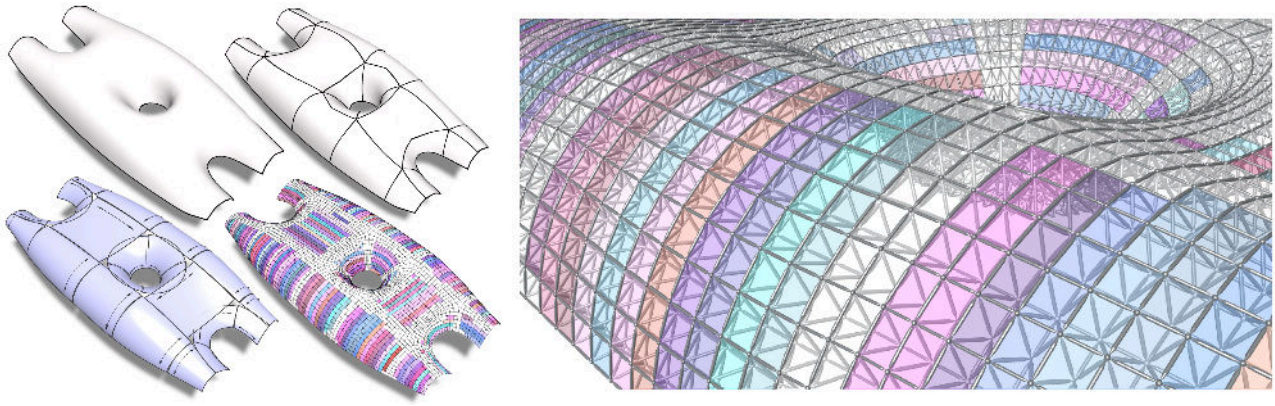


Fig. 1. We approximate a given free-form surface using an assembly of rotational patches. This assembly allows for the creation of a seamless quad mesh that enables the simultaneous repetition of multiple types of building elements. In the presented double-layer space frame structure, all elements, including panels and different layers of nodes and beams, follow the same repetition pattern due to the derived local rotational symmetry. As a result, the number of distinct panels (3,537) is significantly reduced to 1,519 (by 57.1%), with similar reductions for nodes and beams. Different rows of congruent faces are shown in different colors. Transition regions, with faces uniquely shaped to achieve smooth connections across discrete patches, are rendered in plain white.

We present a method to approximate free-form surfaces using assemblies of rotational patches for architectural rationalization. Rotational surface patches inherently allow for the simultaneous repetition of multiple building elements along the arc direction. By assembling multiple patches, we can create diverse free-form-like geometries to satisfy broad design intents, while preserving local symmetry to enable cost-effective element fabrication. The main challenge lies in the strict constraint of maintaining local rotational symmetry, while ensuring the final tessellated form is seamless, smooth, and closely resembles the target surface. To address this, we propose a patch layout creation approach that segments the input surface into patches, resembling untrimmed rotational patches within a prescribed error threshold. Additionally, we develop a B-spline-based optimization framework to refine the fitted rotational patches for smooth connections and faithful surface approximation. To facilitate practical architectural applications, we provide a post-processing tool that converts the discrete patch assembly into a seamless, smooth quad mesh composed of locally repeated elements. We demonstrate that our approach is applicable to a variety of free-form surfaces, including those that mimic iconic architectural designs, and can address various practical requirements for a wide range of application scenarios.

CCS Concepts: • **Computing methodologies → Mesh geometry models;**

Additional Key Words and Phrases: Architectural geometry, rationalization, rotational surface, surface approximation, geometric optimization, fabrication-aware design

ACM Reference Format:
Yuanpeng Liu, Yi Min Xie, Ting-Uei Lee, Ziqi Wang, and Nico Pietroni. 2025. Free-form Surface Approximation Using Rotational Patches. *ACM Trans. Graph.* 44, 5, Article 168 (June 2025), 14 pages. https://doi.org/10.1145/3744707

1 Introduction

Free-form surfaces are increasingly valued in architecture for their innovative and elegant appearances. While advancements in computer-aided design technology have simplified the design process, their intricate geometries, featuring double curvature, still pose significant construction challenges. In response, substantial recent works have focused on *architectural rationalization*, aiming to make the construction of complex surfaces feasible and affordable [Austern et al. 2018; Pottmann et al. 2015].

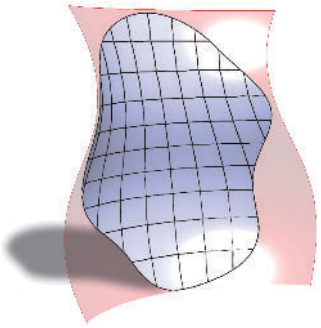
This work was supported by the Australian Research Council (FL190100014).
Authors' Contact Information: Yuanpeng Liu, RMIT University, Melbourne, Victoria, Australia; e-mail: s3875182@student.rmit.edu.au; Yi Min Xie, Hohai University, Changzhou, Jiangsu, China; e-mail: mike.xie@rmit.edu.au; Ting-Uei Lee, RMIT University, Melbourne, Victoria, Australia; e-mail: jeff.lee@rmit.edu.au; Ziqi Wang, The Hong Kong University of Science and Technology, Hong Kong, China; e-mail: ziqiw@ust.hk; Nico Pietroni, University of Technology Sydney, Sydney, New South Wales, Australia; e-mail: nico.pietroni@uts.edu.au.

ACM 0730-0301/2025/06-ART168
https://doi.org/10.1145/3744707

One major rationalization strategy is to reduce the number of distinct shapes for building components such as beams, nodes, and panels. Free-form structures often require numerous unique shapes, leading to costly and time-consuming construction. By limiting the shape variety, mold-based manufacturing methods, like casting, can reuse molds for repeated elements, thereby lowering production costs. For no-mold processes like CNC machining and 3D printing, this strategy simplifies machinery setup and programming, leading to increased efficiency and quality control. Fewer types of different shapes can also ease storage, transportation, and the labeling and identification of elements during on-site assembly.

Previous studies have explored various approaches to reduce the shape variety of building components such as beams [Lee et al. 2022], nodes [Koronaki et al. 2020; Liu et al. 2023], panels [Eigensatz et al. 2010; Fu et al. 2010; Liu et al. 2024, 2021; Pellis et al. 2021; Singh and Schaefer 2010], and so on. These approaches can effectively address individual types of elements, yet real-world architectural projects typically incorporate multiple element types (see Figure 1). Simultaneously reducing the shape variety for each type of element holds great potential for facilitating fabrication but remains under-explored. The main problem lies in optimizing a vast search space that includes both discrete (grouping of individual elements) and continuous (interrelated element geometries) variables. Escaping from sub-optimal local minima remains a significant and ongoing challenge.

In this article, instead of seeking an element-wise solution, we explore surfaces with inherent symmetry that enable simultaneous repetition of multiple elements. Specifically, we focus on rotational surfaces, which are defined by sweeping a 3D profile curve (v) along a circular arc (u) (see Figure 2(a)). By subdividing a rotational surface along the uv directions, the resulting elements are naturally congruent along the generating motion (see Figure 2(b)). Stitching multiple rotational surface patches can further expand the design space to create diverse shapes while maintaining local symmetry for cost-effective fabrication. Based on this insight, the goal of this work is to approximate a given free-form surface using an assembly of rotational patches.



Our rotational patches are subject to three primary design constraints. First, each patch must be an untrimmed, rectangular rotational patch to avoid creating undesired trimmed faces, as shown in the inset figure. Second, patches should closely resemble the target surface, while connecting as seamlessly and smoothly as possible. Third, any residual gaps or kinks between adjacent patches must be addressed in the final tessellated form to enable practical applications. To resolve these constraints, we first segment the input surface into quad patches that resemble untrimmed rotational patches, guided by a customized cross-field. We then optimize the fitted rotational patches for smooth connections and faithful surface approximation. Finally, these discrete patches are tessellated and merged into a seamless quad mesh, with small transition regions introduced to blend out residual errors.

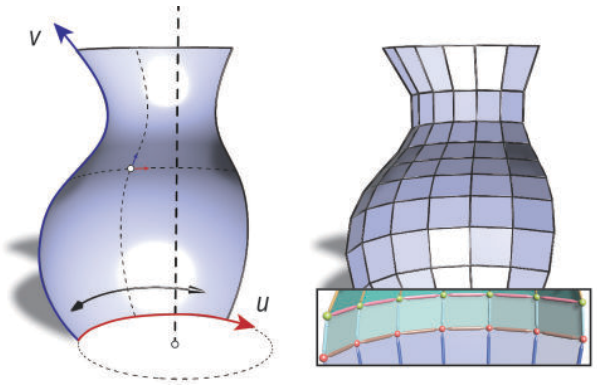


Fig. 2. A rotational surface created by rotating a 3D profile curve (v) along a circular arc (u). By tessellating along uv directions, elements, such as beams, nodes, and panels, along the arc direction are naturally congruent.

Contributions. Our core contributions are summarized as follows:

- We present the first method to approximate a free-form surface using an assembly of untrimmed rotational patches. This approach enables the simultaneous repetition of multiple element types, crucial for cost-effective fabrication.
- We define two new metrics to quantify the resemblance of a surface to a rotational patch. Based on these metrics, we propose a segmentation approach that divides an input surface into quad patches, resembling rotational patches within a specified threshold.
- We develop a B-spline-based global optimization framework to refine the rotational patch assembly for smooth connections, along with other constraints. To facilitate practical applications, we provide a post-processing tool that converts the discrete patch assembly into a continuous quad mesh composed of locally repeated elements.

2 Related Work

In this section, we give a brief review of the methods for surface approximation using discrete classes of equivalent elements, surface approximation using (or segmenting surfaces into) simple primitives, and segmenting surfaces into quadrilateral patches, mainly in the context of architectural applications.

Surface approximation using discrete classes of equivalent elements. Reducing the number of geometrically different elements is a common strategy in architectural applications for cost reduction. Widely considered elements include beams, panels, and nodes. Several methods have been proposed for beams in different contexts, including architectural structures [Bi et al. 2024; Brütting et al. 2021; Lu and Xie 2023], sphere tessellations [Lee et al. 2022; Liu et al. 2022], and small-scale home fabrications [Zimmer and Kobbelt 2014; Zimmer et al. 2014]. As for nodes, methods include replacing low-valence nodes with fewer types of high-valence ones [Bi et al. 2024; Brütting et al. 2021; Zimmer and Kobbelt 2014; Zimmer et al. 2014], or merging different nodes by explicitly achieving congruent shapes [Bo et al. 2011; Koronaki et al. 2020; Liu et al. 2023]. In terms of panels, several methods have been proposed to reduce the number of different molds for panel fabrication, without needing

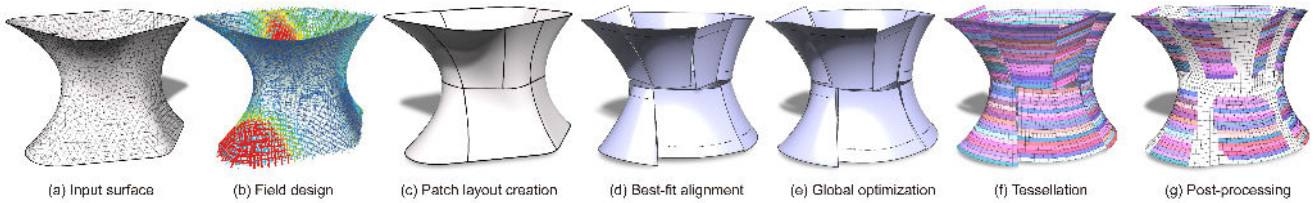


Fig. 3. The overall framework of the algorithm. (a) Initially, we remesh the input surface into an isotropic triangular mesh. (b) We then calculate a smooth cross-field that indicates the best-fit rotational direction at each local region. (c) Next, we trace a set of field-aligned paths to segment the surface into quad sub-meshes that resemble rotational patches within a prescribed threshold. After (d) replacing each sub-mesh with its best-fit rotational patch, (e) we perform a global optimization to refine the appearance of the patch assembly for architectural applications. (f) The optimized patches are then subdivided based on a target edge length while ensuring globally consistent tessellation. (g) Finally, these discrete tessellations are merged, with small transition regions introduced to ensure smooth connections, resulting in a continuous quad mesh composed of locally repeated elements.

the panel shapes to be exactly the same [Eigensatz et al. 2010; Pellis et al. 2021]. Other methods have instead explored approximating free-form surfaces using congruent groups of triangles [Bi et al. 2024; Liu et al. 2021; Singh and Schaefer 2010], quadrilaterals [Fu et al. 2010; Zhu et al. 2023], or general n-gons [Liu et al. 2024]. Other building components include triangle-based point-folding structures [Zimmer et al. 2012] and masonry blocks [Chen et al. 2023], and so on.

Unlike most research that focuses on a single element type, several approaches consider multiple types simultaneously [Bi et al. 2024; Brütting et al. 2021; Zimmer and Kobbelt 2014; Zimmer et al. 2014]. Two methods approximate free-form geometries based on a given set of Zometool beams and nodes, using an advancing-front meshing strategy [Zimmer et al. 2014] or a remeshing approach [Zimmer and Kobbelt 2014]. Brütting et al. [2021] introduces a method to design a set of beams and joints that fit multiple geometries and different structural requirements. Bi et al. [2024] focuses on reducing the number of faces, beams, and nodes in grid shells through clustering and optimization techniques. These methods typically rely on high-valence nodes to provide additional connection options to enable the reuse of the same shape in multiple positions. In contrast, our approach explicitly achieves congruent node geometries without increasing node valence. Moreover, our method can be applied to multiple and various elements that can be assembled based on a rotational surface, including but not limited to beams, panels, and nodes.

Instead of explicitly adjusting the element shape, some studies tackle the problem from a surface perspective, seeking geometries with inherent properties that implicitly facilitate element repetition. For instance, the approach described in Pellis et al. [2021] transforms a given free-form surface to a close-by Weigartern surface, allowing the use of the same mold for panels aligned along the curvature-isolines, without needing the panels to be congruent. Similarly, our approach also adopts a surface-based strategy. However, we focus on rotational surfaces, which produce exactly congruent elements and can be applied to multiple element types.

Surface approximation using simple primitives. The task of surface approximation using simple shapes has been widely studied across multiple fields, including design rationalization, mesh simplification, reverse engineering, and so on. Depending on the intended geometry, commonly investigated shapes include planes [Chen et al. 2013; Cohen-Steiner et al. 2004; Jadon et al. 2022; Liu et al. 2006b, 2011; Pluta et al. 2021; Wu and Kobbelt 2005], spheres [Jadon et al. 2022; Wu and Kobbelt 2005], quadrics [Yan et al. 2012], cylinders [Jadon et al. 2022; Wu and Kobbelt 2005], developable surfaces [Stein et al. 2018; Zhao et al. 2023], ellipsoids [Simari and Singh 2005; Yan et al. 2012], and so on. These methods generally involve a segmentation process that splits the given surface into patches approximating the target shape, followed by a fine-tuning stage to refine the geometry. However, most approaches generate unstructured patches with complex and irregular boundaries that hinder proper tessellation. While some studies successfully achieve regular tessellation of surfaces with planar faces through specialized meshing techniques [Liu et al. 2006b, 2011; Pluta et al. 2021], these methods are specifically designed for planar faces and are less adaptable to other shaped patches. Regarding general rotational patches, existing studies are mainly focused on fitting a single rotational patch to the given geometry [Liu et al. 2006a; Pottmann and Randrup 1998; Pottmann and Wallner 2001]. In this study, we present the first method to approximate free-form surfaces using an assembly of untrimmed (rectangular) rotational patches, which can be tessellated into repetitive elements, holding great potential for reducing fabrication costs.

Partitioning Surfaces into Quadrilateral Patches. Our method deals with untrimmed rotational patches that are inherently quadrilateral, necessitating the segmentation of the input surface into a quadrilateral patch layout. Various methods for creating quad layouts have been proposed for different purposes, such as semi-regular quad remeshing [Bommes et al. 2013] and globally smooth parameterization [Myles et al. 2014] (see [Campen 2017] for a comprehensive survey). Our approach is similar to patch decomposition techniques for quadrilateral remeshing, both aiming to subdivide each patch to achieve smooth and consistent tessellation. However, unlike existing approaches, we pose additional demands on the patch shape, requiring each patch to resemble a rotational patch within a prescribed threshold.

An effective strategy for constructing a quad patch layout is to rely on an auxiliary tangent space direction field to control the orientation of the edges. Principal patches, with edges following principal curvature lines, have been investigated in discrete differential geometry, offering possibilities to construct C^1 surfaces from

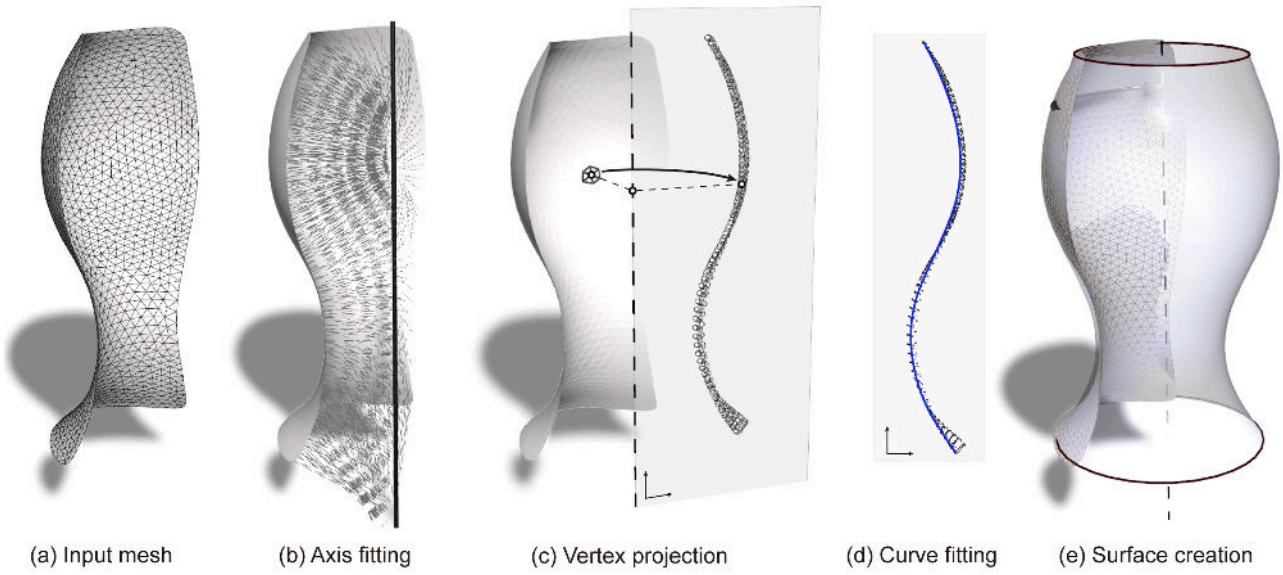


Fig. 4. To calculate δ_g of an input mesh (a), we begin by fitting an axis to be as co-planar as possible to the normal lines of mesh faces (b). Based on this axis, we rotate all mesh vertices onto a half-plane (c). Next, we fit a profile curve to the resulting planar point set (d), and subsequently construct the fitted rotational surface (e).

patches of Dupin cyclides [Bo et al. 2011; Bobenko and Huhnen-Venedey 2012; Martin 1982]. State-of-the-art techniques compute a smooth tangent vector field on the surface [Vaxman et al. 2017] aligned to features and principal directions as the basis for tracing field-aligned paths to construct the patch decomposition for various applications [Livesu et al. 2020; Nuvoli et al. 2019; Pietroni et al. 2022, 2021, 2016; Razafindrazaka et al. 2015]. Significantly, Pietroni et al. [2022] further integrates geometric constraints on the patch shape during segmentation to facilitate garment fabrication. In this study, we adapt the framework by Pietroni et al. [2022] to suit rotational patches while addressing different requirements on topology, field, and patch shape. Specifically, in our approach, all patches must be rectangular, with T-junctions being allowed. A field is specifically designed to indicate the local rotational direction to guide path tracing. Patch shape is constrained within an error threshold measuring the resemblance of a given mesh to its best-fit rotational patch.

3 Methodology

The overview of our method is outlined in Figure 3. Two similarity metrics are defined in Section 3.1 as the basis of our algorithm. Given an input surface (Figure 3(a)), we first calculate a smooth cross-field (Figure 3(b)) that aligns with the fitted rotational direction at each local region, as described in Section 3.2. Next, we trace field-aligned paths to segment the surface into quad patches (Figure 3(c)) that resemble rotational patches within a prescribed error threshold, as explained in Section 3.3. These patches are then replaced with their best-fit rotational patches (Figure 3(d)), further refined via global optimization (outlined in Section 3.4) for smooth connections and faithful surface approximation (Figure 3(e)). Finally, the optimized patches are tessellated according to a target length (Figure 3(f)), as described in Section 3.5, and further converted into a continuous quad mesh (Figure 3(g)) to facilitate practical applications, as detailed in Section 3.6.

3.1 Similarity Metric

We first define two metrics, δ_g and δ_b, to quantify the similarity of a mesh to a rotational patch. The first metric δ_g measures the distances from mesh vertices to a rotational surface built from a fitted rotational axis and profile curve. It reflects the general conformance of the mesh shape, regardless of its boundaries. In contrast, the second metric δ_b is more stringent, involving optimization over the rotational patch based on sample points to ensure precise alignment with the mesh, including both interior shape and boundaries.

3.1.1 δ_g, General Fit.

Fitting the rotational axis. To fit a given mesh by a rotational surface, we first determine the rotational axis using the method described in Pottmann and Randrup [1998] and Pottmann and Wallner [2001]. The core idea is to find the axis as a line that is as co-planar as possible with the normal lines of the mesh faces (see Figure 4(b)), where each normal line passes through the face center along the normal direction. In this approach, a line is represented in Plücker coordinates, expressed as a six-element vector $(\boldsymbol{l}, \bar{\boldsymbol{l}})$, where $\boldsymbol{l}$ is a normalized direction vector and $\bar{\boldsymbol{l}} = \boldsymbol{p} \times \boldsymbol{l}$ is a moment vector, with $\boldsymbol{p}$ denoting the coordinate vector of an arbitrary point on this line. The coplanarity of two lines $(\boldsymbol{a}, \bar{\boldsymbol{a}})$ and $(\boldsymbol{b}, \bar{\boldsymbol{b}})$ can then be conveniently expressed as $\boldsymbol{a} \cdot \bar{\boldsymbol{b}} + \boldsymbol{b} \cdot \bar{\boldsymbol{a}} = 0$. For the set of normal lines from the mesh faces $(\boldsymbol{n}_i, \bar{\boldsymbol{n}}_i)$, we seek a line $(\boldsymbol{x}, \bar{\boldsymbol{x}})$ that minimizes the coplanarity measure, as formulated in Equation (1), while satisfying the normalization condition in Equation (2) and the Plücker condition in Equation (3) to ensure

a valid line representation. We first omit the second constraint and solve the subsystem by converting it into a general eigenvalue problem, where the solution is the eigenvector associated with the smallest non-negative eigenvalue, as described in Pottmann and Wallner [2001] and Pottmann and Randrup [1998]. We then take this solution as initialization for solving the full system. To enforce the nonlinear constraints, we use a penalty method, integrating squared constraint terms into the objective function with a large penalty weight (set to 1,000). The full system is then optimized using a quasi-Newton solver, which typically converges within a few iterations.

$$\min \quad F_{axis} = \sum (\boldsymbol{n_i} \cdot \bar{\boldsymbol{x}} + \boldsymbol{x} \cdot \bar{\boldsymbol{n}}_{\boldsymbol{i}})^2, \tag{1}$$

$$\text{s.t.} \quad ||\boldsymbol{x}||^2 = 1, \tag{2}$$

$$x \cdot \bar{x} = 0. \tag{3}$$

Fitting the profile curve. After determining the axis, all mesh vertices are rotated onto a half-plane, forming a planar point set that defines the profile shape (see Figure 4(c)). We then fit the profile curve (modeled as a uniform cubic B-spline curve) to this point set via optimization, as shown in Figure 4(d). The objective function $F_{B-spline}$ is specified in Equation (4). It includes an error term e_{dk} that reflects the distance from each data point to the curve; here, the curvature-based distance term proposed in Wang et al. [2006] is adopted for efficiency, formulated in the appendix (Equation (13)). Since the fitted curve serves as the profile curve of a rotational surface, curve fold-over or self-intersection is prohibited. To address this, we add a term e_{nk} considering the normal direction (Equation (5)), where N_k denotes the projected normal of each mesh vertex and T_k denotes the tangent vector at the closest point on the B-spline curve. The effect of this term is illustrated in Figure 5. F_s is a smoothness term (Equation (6)) that regulates the first and second derivatives $P'(t)$ and $P''(t)$ of the B-spline curve $P(t)$. The weights for these terms are set as $w_d = w_n = 1$ and $w_s = 0.01$, respectively.

During optimization, the control points of the B-spline curve serve as the design variables, with their number set to eight by default. The optimization is solved iteratively. In each iteration, we first update the closest point of each data point on the current B-spline curve, assigning each data point a fixed parametric value t_k. In the distance term e_{dk}, the point position $P(t_k)$ has a linear relationship with the control points, while other variables are derived from the current geometry and treated as constant. In the normal term e_{nk}, the normal N_k is derived from the input mesh and remains fixed, while the tangent vector T_k has a linear relationship with control points. The integrals within the smoothness term F_s are computed analytically. Since $P(t)$ is linear in control points, its derivatives $P'(t)$ and $P''(t)$ are also linear. As a result, in each iteration, the optimization problem reduces to a quadratic program, which is solved by setting the first derivative to zero and solving the resulting linear system.

Eventually, the similarity metric δ_g is computed as the sum of squared distances between each data point and its closest position on the optimized B-spline curve, normalized by the number of data points.

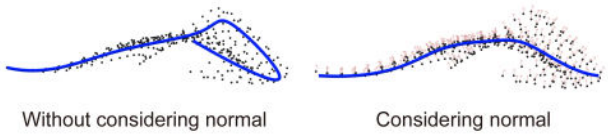


Fig. 5. Compared to using the distance term alone, adding the normal term helps to prevent curve folding-over and self-intersection.

$$\min \quad F_{B-spline} = \sum_{k=1}^{n} (w_d e_{dk} + w_n e_{nk}) + w_s F_s, \tag{4}$$

$$e_{nk} = (T_k^T \cdot N_k)^2, \tag{5}$$

$$F_s = \int \|P'(t)\|^2 \, dt + \int \|P''(t)\|^2 \, dt. \tag{6}$$

By rotating the identified profile curve around the fitted axis, we can generate a rotational surface, as shown in Figure 4(e). This two-step fitting approach can be applied to any mesh, regardless of its boundary, and efficiently produces a good approximation of a rotational surface. Although the precision of this approach can be further improved via additional optimization, as explored in Liu et al. [2006a], here efficiency is prioritized over precision. This is because this approach is utilized in the field design stage to determine the local rotational directions over the entire surface (discussed in Section 3.2), where it may be processed thousands of times, and the field mainly serves as guidance for path tracing. In this regard, we adopt the unmodified, more efficient approach in our implementation.

3.1.2 δ_b, Including Boundary Alignment.

Optimization formulation. To evaluate the similarity of a given mesh to a rotational patch while considering the alignment of both the interior shape and the boundary, we perform optimization to determine the best-fit untrimmed rotational patch. Specifically, we uniformly sample a grid of test points $P(u, v)$ on the patch and seek to minimize their distances to the corresponding closest points P^*_{uv} on the mesh, as shown in Figure 6. To ensure alignment with the mesh boundaries, corner points on the patch are matched to their closest mesh corners, and boundary points are aligned with their nearest positions on the corresponding mesh boundaries. The objective function is detailed in Equation (7), where the control points of the profile curve (represented by a uniform cubic B-spline) P_b and the arc P_a are taken as the design variables. The number of controls points for the profile curve is set to four for efficiency, considering that the optimization problem is highly nonlinear and architectural surfaces generally exhibit low local curvature variations. The sampling densities along the u and v directions are denoted by n_u and n_v, respectively. The optimization problem is solved using a quasi-Newton solver. The target positions P^*_{uv} are updated at the beginning of each iteration and treated as constants.

$$\min \quad F_{patch} = \sum_{i=0}^{n_u} \sum_{j=0}^{n_v} ||P\left(\frac{i}{n_u}, \frac{j}{n_v}\right) - P^*_{uv}||^2 + w_s F_s. \tag{7}$$

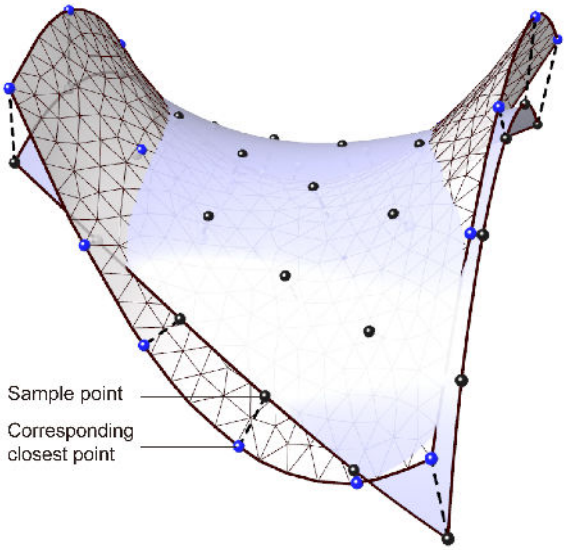


Fig. 6. To calculate δ_b, we uniformly sample a grid of test points on the rotational patch and seek their corresponding closest points on the mesh. Corner points on the patch are matched to their closest mesh corners, and boundary points are aligned with their nearest positions on the corresponding mesh boundaries.

For a rectangular mesh patch, there are two possible rotational directions, with either one pair of opposite edges as arcs and the other as profile curves. We evaluate both configurations and select one with the lower error to calculate δ_b, which equals the sum of squared distances from sample points to their target positions on the input mesh, normalized by the number of sample points. This metric is utilized in the mesh segmentation process to evaluate the distance from a sub-mesh to its best-fit untrimmed rotational patch, as further discussed in Section 3.3.

3.2 Cross-field Construction

Given an input mesh, we compute a vector field to guide the path tracing for patch layout decomposition. Since each patch must be quadrilateral and it is preferred to have orthogonal patch corners for creating well-shaped panels, a cross-field (4-RoSy tangent-vector field [Vaxman et al. 2017]) is selected for the guidance. As we seek to segment the mesh into sub-meshes that resemble rotational patches, we compute a field that aligns with local rotational directions to capture the intrinsic rotational symmetries. To construct such a cross-field, we first calculate a unit vector at each face following the orthogonal projection of a fitted rotational axis of the local region. Each direction vector is then weighted according to the resemblance of each local region to a rotational surface and the uniqueness of the axis. The final cross-field can then be computed through global smoothing by taking these weighted direction vectors as soft constraints.

Direction vector. For each face, we compute the direction vector by fitting a rotational axis to its local region (see Section 3.1.1). The local region is defined as the largest N-ring neighbors ($N_{\max}$-ring) around this face that approximates a rotational surface within a prescribed threshold (measured relative to δ_g). The final direction vector is obtained by projecting the fitted rotational axis onto the corresponding mesh face.

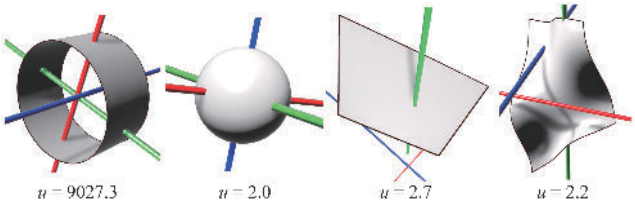


Fig. 7. Geometries with unique axes, such as cylinders, present much larger uniqueness value u compared to spheres or planes, which possess infinite possible rotational axes. Arbitrary surfaces with little rotational symmetry in any direction also contain small u.

Weight. We assign a weight to each face to distinguish regions with strong rotational characteristics i.e., those that closely resemble rotational surfaces and exhibit a unique rotational axis. The assigned weight influences the subsequent field smoothing process, where higher-weighted direction vectors are better preserved, while those in less significant regions are smoothed out. The weight w is computed as $w = N_{\max}u$. Here, $N_{\max}$ represents the size of the local region for each face (previously defined during direction vector computation); a larger $N_{\max}$ typically indicates a stronger resemblance to a rotational surface. The term u quantifies the uniqueness of the best-fit rotational axis within the local region, indicating how sensitive the fitting error is to variations in axis position and orientation (further detailed below). Higher weights indicate stronger rotational characteristics and are illustrated in warmer colors in Figure 8(b).

Uniqueness of the rotational axis. Regions with a unique rotational axis (e.g., cylinders) are prioritized over those with multiple potential rotational axes (e.g., planes and spheres). To compute the uniqueness measure u for each local region, we derive three "typical" axes from the eigenvectors of the matrix formulation of Equations (1) and (2). This 6×6 matrix contains three non-trivial (non-zero) eigenvalues, whose corresponding eigenvectors represent the Plücker coordinates of lines (see [Pottmann and Randrup 1998] for more details on matrix construction and analysis). These eigenvectors are further optimized to satisfy the Plücker condition (Equation (3)), ensuring valid line representations. The three extracted lines are treated as potential rotational axes (the one corresponding to the minimal eigenvalue is the best-fit rotational axis, defined in Section 3.1.1), and for each, a rotational surface is fitted to compute the corresponding fitting error (similar to Equation (4)). The uniqueness measure u is then defined as the ratio of the largest to smallest fitting error, quantifying the sensitivity of the fitting error to axis variations. As illustrated in Figure 7, cylindrical regions exhibit one small and two large fitting errors, resulting in a large u. Planar and spherical regions have small fitting errors for all axes, resulting in a low u. Arbitrary surfaces with little rotational symmetry exhibit three large fitting errors, also producing a low u. Regions with higher sensitivity to variations in rotational axes (higher u) receive larger weights to better preserve their rotational directions during the subsequent smoothing process.

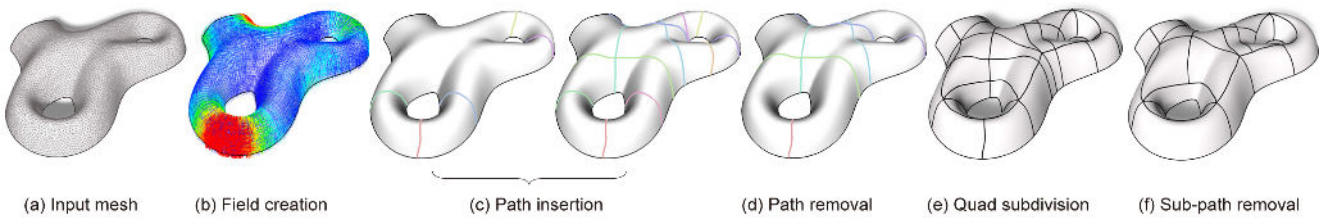


Fig. 8. The framework for patch layout creation. Given an input mesh (a), we first calculate a smooth cross-field that indicates the local rotational directions, where warmer colors indicate larger weights (b). We then gradually insert field-aligned paths to split the surface into finer, uniform patches, until all satisfy the given constraints (c). Unnecessary paths are removed to simplify the patch layout (d). Next, we split non-quad patches via subdivision to create a pure-quad patch layout (e). Eventually, redundant sub-paths (i.e., patch edges) are further removed if the merged patches still meet the given constraints, resulting in the final quad patch layout (f).

Cross-field smoothing. With the obtained direction vectors and weights at each face (or a subset of sampled faces for efficiency), we take these weighted vectors as soft constraints for cross-field smoothing. To ensure patches near the boundaries achieve a proper shape with near orthogonal corners, we further require the field to align with the boundary edges, such that the field-aligned paths intersect the boundaries orthogonally. The final smoothed cross-field, adhering to these constraints, can be computed using the method described in Diamanti et al. [2014].

3.3 Patch Layout Construction

Overview. The next step involves constructing a patch layout that follows the computed cross-field to segment the surface into rectangular sub-meshes, resembling untrimmed rotational patches within a specified threshold ϵ_s (measured with respect to δ_b). The patch layout creation process is summarized in Figure 8. We first gradually insert field-aligned paths across the surface to split it into uniform, finer patches, ensuring each conforms to the set constraints (see Figure 8(c)). Since patch shapes depend on all sequentially added paths, some earlier paths may become redundant as new ones are introduced. To simplify the layout, we identify and remove these unnecessary paths (see Figure 8(d)). Next, we perform quad subdivision to convert all non-quad patches into quadrilateral ones (see Figure 8(e)). Finally, the quad layout is further refined by removing sub-paths (i.e., patch edges) where adjacent patches can be merged without violating the set constraints.

Single path tracing. To trace a field-oriented path, we employ the graph-based approach described in Nuvoli et al. [2019] and Pietroni et al. [2021, 2016]. Initially, we interpolate the cross-field at mesh vertices, derived from the surface-based cross-field, while ensuring 4-Rosy symmetry. We create a graph with four nodes at each mesh vertex, one for each direction of its cross-field, as shown in Figure 9. We then connect adjacent nodes with matching cross-field directions, and assign each connection a weight based on its deviation from the intended direction at adjacent nodes. Based on this setup, path tracing can be cast as a shortest-path search problem from a given source node to a destination node within a weighted graph.

Two types of paths are crucial for creating a valid patch layout: border-to-border paths and self-connecting loops. For loops, we designate a single vertex node as both the source and the destination, ensuring the path forms a closed circuit. For border-to-border paths, we select source nodes on boundary vertices where the direction orthogonally enters the mesh, and destination nodes where the direction orthogonally exits, thus avoiding ill-shaped patches intersecting with the boundary at sharp angles. This orthogonality is ensured by aligning the field with the boundary edges (see Section 3.2). Since paths are sequences of adjacent edges from a triangle mesh, they are typically in zigzag shapes. We smooth these paths, reproject them over the surface to enhance the appearance, and update the rest of the mesh accordingly.

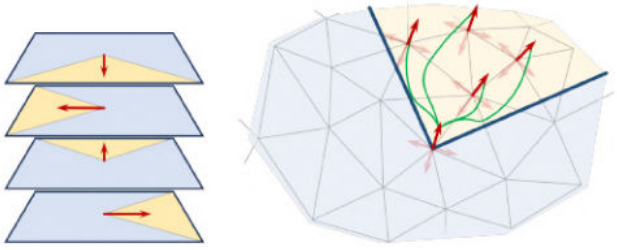


Fig. 9. We create a graph with four nodes at each mesh vertex, one for each direction of its cross-field, and then connect adjacent nodes with matching cross-field directions.

Path insertion for layout creation. Initially, we sample a set of candidate paths by tracing both border-to-border paths and loops. To enhance efficiency, we uniformly subsample source nodes located on the border and within the interior of the mesh. Similarly to Livesu et al. [2020], we insert a path using a greedy strategy that prioritizes the furthest path from the previously inserted one. As in Pietroni et al. [2021, 2016], the distance is computed for each node using the M4 stratification of the graph [Campen et al. 2012], and averaged for each path. Note that two paths crossing orthogonally can be very far, and in contrast, parallel paths are typically close. This simple strategy prevents dense gathering of paths and tends to create a uniform patch layout decomposition.

The patch layout is continually updated as each path is iteratively added. At the start of each iteration, we assess the current patches to determine if they meet our objectives. Quad patches are directly checked against the given error threshold. For non-quad patches, we check if they can be split into quad sub-patches (via quad subdivision, detailed in the following paragraph), with all resulting sub-patches meeting the threshold. Note that during this

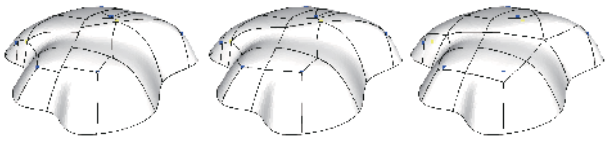


Fig. 10. Non-quadrilateral patches are subdivided by first connecting the internal singularities to selected split points on each side (left). Next, the split points along the sides are averaged across adjacent patches (center). Finally, the central point of each patch is optimized based on the averaged split points (right).

evaluation, the patch layout remains unchanged, and the splitting is performed virtually for error measurement. If all patches meet our objectives, the iteration stops. Otherwise, we insert a new path from the candidate pool that can divide any unsatisfactory patch without tangentially intersecting any existing path (see Figure 8(c)).

Quad subdivision. The splitting process is performed by inserting new paths from field singularities towards each side of the patch. Each non-quad patch is first mapped to 2D by assigning its boundary to a convex regular polygon and parametrizing the internal vertices using least-squares conformal maps [Lévy et al. 2002]. In this 2D space, the polygon is divided into quads by connecting the internal singularity to each polygon side. For each side, the splitting point is chosen to create angles that are as orthogonal as possible to the boundary in the parametric space (see Figure 10, left). However, this step can introduce unnecessary T-junctions in the patch layout because adjacent splitting points are generally misaligned. To address this, a second step averages the splitting points of adjacent patches along shared edges, simplifying the layout by removing these T-junctions (see Figure 10, center). Finally, the central point (centroid) of each patch is repositioned to further enhance the orthogonality of angles in the generated quads (see Figure 10, right). This is done by first mapping the patch to 2D using as-rigid-as-possible parametrization [Liu et al. 2008], and then updating the centroid to the mesh vertex that minimizes the sum of squared deviations of quad angles from 90 degrees. With the updated centroids, splitting points at patch sides are refined by selecting the closest vertex (in terms of geodesic distance) to the adjacent centroids. The centroids and splitting points are iteratively adjusted, with experimental results showing that five iterations are typically sufficient for convergence.

Once the path insertion is complete, we obtain a patch layout that meets our objectives. However, during the construction process, it is uncertain whether a candidate path will be essential in the final layout. As patch shapes evolve with subsequently added paths, some earlier paths may become redundant as new ones are introduced. We further check and remove redundant paths to simplify the patch layout (see Figure 8(d)). This involves checking each path in reverse order, starting from the last inserted one, and removing it if all merged patches along this path still meet the required threshold. Based on the simplified layout, we split all non-quad patches via quad subdivision to create a pure-quad patch layout (see Figure 8(e)), incorporating T-junctions as needed. Afterwards, we assess each sub-path (i.e., patch edge) and remove it if its adjacent patches can be merged without exceeding the threshold or introducing a non-quad patch. This iterative process merges patches with the smallest error first, eventually resulting in the final simplified, pure-quad patch layout (see Figure 8(f)).

This patch layout creation process does not guarantee convergence for arbitrarily low error thresholds. When the threshold is too strict, it essentially reduces to segmenting the input surface into perfect rotational patches, which is generally infeasible. However, since the primary goal at this stage is to generate a rough approximation for initializing the subsequent global optimization, a relatively higher error threshold is typically chosen.

3.4 Global Optimization

After surface segmentation, we replace each sub-mesh with its best-fit untrimmed rotational patch (see Figure 3(d)). To refine this patch assembly for architectural applications, we conduct global optimization, with multiple constraints detailed as follows.

Rotational patch (hard constraint). Throughout optimization, all patches are strictly kept as untrimmed rotational patches. This is achieved by taking the control points of the profile curves and arcs as design variables. Since all patches are built from the rotation of curves, they always retain their rotational form regardless of curve shapes. This ensures that the subdivided elements along the arc direction in the final tessellated form are exactly congruent.

Closeness, E_c (soft constraint). To ensure close surface approximation, we sample test points on each patch and minimize their distances to corresponding target positions on the input surface (as in Section 3.1.2). The closeness term E_c is defined as the sum of F_{patch} (see Equation (7)) over all patches.

Gap closure, E_g (soft constraint). Adjacent patches should connect as seamlessly as possible. For each pair of adjacent edges, we uniformly sample n_s points along each edge and minimize the sum of squared distances between corresponding vertex pairs (see Figure 11). The detailed formulation is given in Equation (8), where n_E denotes the number of edge pairs, V_1 and V_2 denote the sample points along each edge, and i and j are the indices of patch and sample point, respectively.

$$E_g = \sum_{i=1}^{n_E} \sum_{j=0}^{n_s} ||V_{1ij} - V_{2ij}||^2. \tag{8}$$

Surface smoothness, E_n (soft constraint). To achieve smooth surface connection across patches, for each pair of adjacent edges, we require the unitized normal vectors N_1 and N_2 at sample points (the same points used in E_g, see Figure 11) to be aligned, and also orthogonal to the line connecting the corresponding sample points V_1V_2 (denoted by v, the unit vector of $V_1 - V_2$). The detailed formulation is given in Equation (9), where i and j denote the indices of patches and sample points, respectively.

$$E_n = \sum_{i=1}^{n_E} \sum_{j=0}^{n_s} ((1 - (N_{1ij} \cdot N_{2ij})^2)^2 + (N_{1ij} \cdot v_{ij})^4 + (N_{2ij} \cdot v_{ij})^4). \tag{9}$$

Edge smoothness, E_t (soft constraint). To achieve smooth edge transition across patches, we require the tangent vectors T_1 and T_2 at the endpoints of adjacent edges (see Figure 11) to be aligned and collinear with the line connecting the corresponding sample

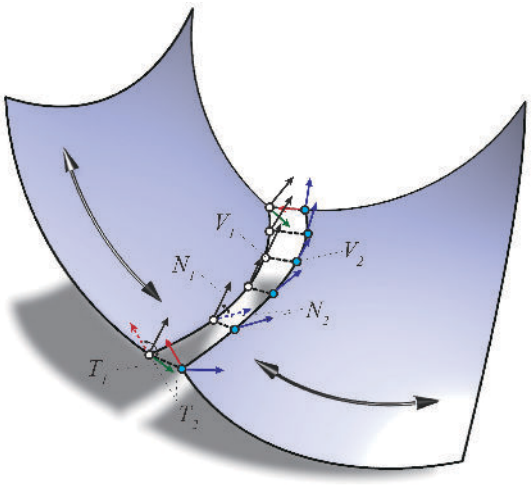


Fig. 11. During optimization, several soft constraints are included for adjacent patches to merge gaps (V_1 and V_2), ensure smooth surface transitions (N_1 and N_2), and achieve smooth curve transitions for patch edges (T_1 and T_2).

points (denoted by v), as formulated in Equation (10). Here, i and j denote the indices of patches and endpoints, respectively. This term is only applied to boundary and valence-4 vertices, ignored for singularities, where the non-smoothness of a tessellation is typically concentrated.

$$E_t = \sum_{i=1}^{n_E} \sum_{j=1}^{2} ((1 - (T_{1ij} \cdot T_{2ij})^2)^2 + ((1 - (T_{1ij} \cdot v_{ij})^2)^2 + ((1 - (T_{2ij} \cdot v_{ij})^2)^2). \quad (10)$$

Global objective function. Summing up the weighted constraints gives the global objective function, as formulated in Equation (11). The sample densities n_u, n_v, n_s are all set to 5. The values of weights $\{w_c, w_g, w_n, w_t\}$ are set empirically to $\{10, 1, 0.1, 0.01\}$, unless otherwise specified. Users are free to explore different weight combinations for practical usage. For E_g, E_n, and E_t, at T-junctions, we split the long edge into two segments and follow the same procedure.

$$E_{total} = w_c E_c + w_g E_g + w_n E_n + w_t E_t. \quad (11)$$

3.5 Tessellation

Based on the optimized patch assembly, we calculate the edge subdivision numbers to uniformly tessellate patches along the *uv* directions. To achieve a globally consistent tessellation, opposite edges within each patch and adjacent edges of neighboring patches must have the same subdivision numbers. At T-junctions, the number of the long edge equals the sum of the two short edges. Based on these criteria, we group edges based on the patch layout topology, with edges per group sharing the same subdivision number. Long edges at T-junctions are split into two for grouping. Eventually, each group is assigned a subdivision number as the nearest integer to $\frac{\sum l_i}{n_e l^*}$, where l_i is the length of the ith edge and n_e is the number of edges in the group, and l^* is the target edge length for the subdivided panel.

3.6 Post-processing

For general input surfaces, the multiple constraints in global optimization typically conflict, leaving residual gaps and kinks between adjacent patches that are undesirable for practical applications. To resolve this, we propose a post-processing method that converts the discrete patch tessellations into a seamless, smooth quad mesh, as illustrated in Figure 12. The key idea is to relax the rotational symmetry near patch boundaries by assigning certain mesh faces as transition regions, allowing them to deform for smooth connections. Specifically, the transition regions are gradually expanded based on a specified error threshold. Third-order Laplacian smoothing [Botsch and Kobbelt 2004] is applied on these transition regions to ensure a C^2 smooth blend with fixed regions. This process eventually yields a continuous mesh with locally repeated elements, while elements within the transition regions typically exhibit distinct shapes. The main steps are detailed as follows:

Step 1 *Merge adjacent patches.* Given the discrete tessellations, adjust each boundary vertex to the average position of corresponding vertices from adjacent patches, then merge them. These merged vertices form the initial list of free vertices.

Step 2 *Smoothing.* Optimize the positions of free vertices using third-order Laplacian smoothing, while keeping the remaining vertices fixed. Uniform weighting is applied to smooth the tessellation simultaneously.

Step 3 *Update transition region.* For each fixed vertex adjacent to a free vertex, calculate its distance to the average position of its neighboring vertices. If the distance exceeds a specified threshold, add this vertex to the list of free vertices.

Step 4 *Termination check.* Terminate the iteration when all examined vertices meet the threshold, or the maximum number of iterations is reached. Otherwise, return to Step 2.

4 Results and Discussion

Given an input free-form surface, our method produces an assembly of closely aligned rotational patches, further converted into a continuous quad mesh with locally repeated elements, mainly serving for rationalizing architectural surface designs. The algorithm is implemented in C++ based on Libigl [Jacobson et al. 2018] and C# within the Rhino-Grasshopper platform, on a desktop computer with a 5.1GHz CPU and 32GB memory. Optimization problems are solved using the L-BFGS algorithm in Alglib [Bochkanov 2013] and auto-differentiation in FADBAD++ [Bendtsen 1996]. Our approach is tested across a variety of free-form surfaces, including those that mimic iconic architectural designs, as shown in Figures 13–22, with the statistics summarized in Table 1. Each input surface is normalized such that the longest edge of its bounding box equals one. The error thresholds for patch layout creation (ϵ_s, see Section 3.3) and for smoothing during post-processing (δ_s, see Section 3.6) are set to 0.0025 and 0.05 by default, unless otherwise specified. For each example, we show the input mesh with the generated patch layout, the optimized patch assembly, and the final quad mesh after post-processing. For simplicity, we only record the statistics of faces; the numbers of distinct shapes for other elements like beams and nodes are typically reduced by a similar ratio.

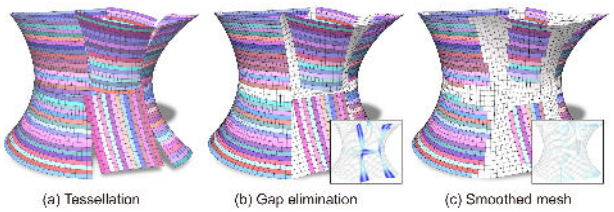


Fig. 12. Post-processing. Based on the tessellations of discrete rotational patches (a), we first directly merge adjacent patches, resulting in bumpy connections (b). The bottom sub-figure shows the error distribution, indicating the distance from each vertex to the average position of its neighbors, with darker colors representing larger errors. By iteratively expanding the transition region and smoothing, we achieve a final seamless, smooth mesh with significantly reduced errors (c).

4.1 Architectural Application Considerations

For architectural applications, the generated design should closely approximate the target surface, exhibit a seamless, smooth appearance, and enable cost-effective fabrication. Additional considerations may include preserving surface boundaries, maintaining global symmetry, controlling panel dimensions, and designing the actual paneling of the surface. Below, we demonstrate how our method addresses these various requirements, making it applicable for a wide range of application scenarios.

Closeness. To achieve a closer approximation of the target surface, we provide two strategies. First, one can increase the weight of the closeness term (ω_c, see Section 3.4) for optimization, as shown in Figure 13. This can effectively produce a closer approximation, but introduces larger gaps between patches, requiring wider transition regions and more distinct element shapes. Additionally, one could also use a smaller error threshold (ϵ_s, see Section 3.3) for segmentation, creating more patches for approximation, as shown in Figure 14. A larger number of patches provides more degrees of freedom for a closer fit, but also increases the size of transition regions and results in more distinct shapes. Balancing cost-effectiveness and approximation is a design decision, depending on the specific project requirements.

Smoothness. Our method allows users to control the smoothness of the resulting mesh by adjusting the error threshold (δ_s, see Section 3.6) for post-processing, as shown in Figure 15. Generally, a smaller δ_s produces a smoother mesh, yet also results in wider transition regions and more uniquely shaped elements.

Enhancing cost-effectiveness. Despite the trade-off between closeness, smoothness, and cost-effectiveness, we propose an iterative strategy that can slightly reduce the number of distinct elements without largely altering the surface shape. This is achieved by iteratively setting the smoothed geometry after post-processing as the new target surface and performing additional optimization. As illustrated in Figure 16, this approach further reduces the number of distinct faces by 4.8%, with the overall geometry remaining nearly unchanged.

Boundary and symmetry constraints. Our algorithm can preserve boundary features by assigning larger weights to corresponding vertices in the closeness term, as shown in Figure 17. Global

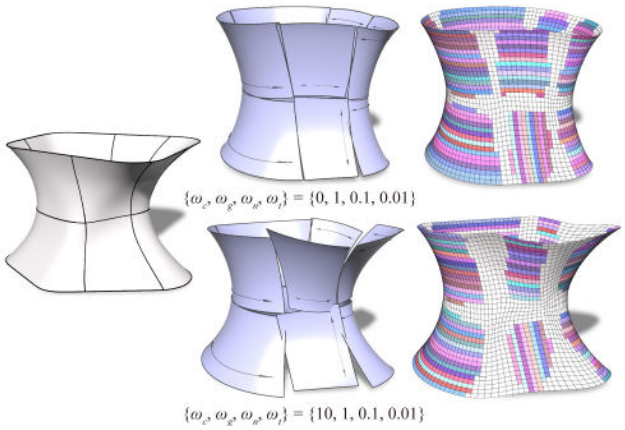


Fig. 13. Increasing the weight of the closeness term (ω_c) allows for a closer approximation of the target shape but results in larger gaps, requiring wider transition regions and more distinct element shapes.

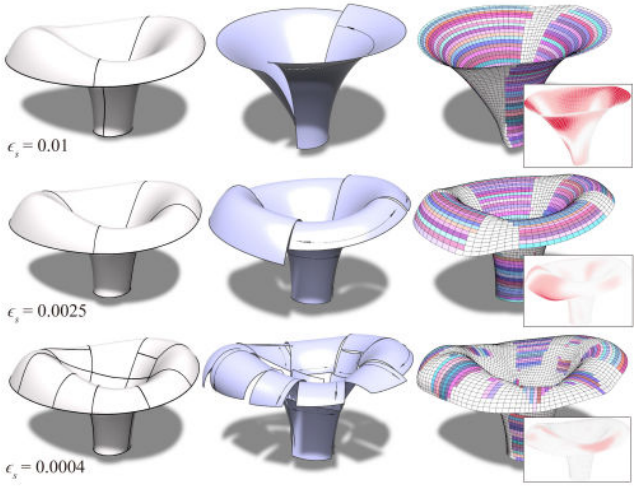


Fig. 14. Reducing the error threshold ϵ_s results in segmentation with a greater number of patches, enabling a closer approximation to the input surface. The sub-figures at the bottom right illustrate the error distribution of deviations from mesh vertices to the input surface, with darker colors indicating larger deviations.

symmetry can also be maintained within our framework, throughout the generated patch layout, optimized patch assembly, and final continuous quad mesh, as illustrated in Figure 18.

Panel planarity. Our method supports the integration of additional constraints, such as face planarity. In Figure 20, we further require the B-spline curve to be coplanar with the rotational axis by adding the term from Equation (12) into the objective function. Here, V_i denotes the volume of the tetrahedron formed by the ith and (i+1)th B-spline control points and two points on the rotational axis. After optimization, we obtain faces with improved planarity. The derived mesh is both *conical* and *circular* (except for the transition regions), which contains desirable features for architectural applications, such as planar faces, torsion-free nodes, and offset

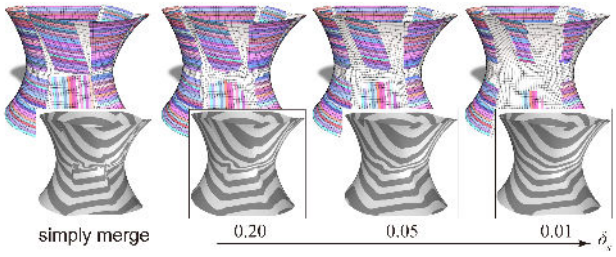


Fig. 15. Smoothness can be controlled by adjusting the error threshold δ_s. A lower value results in a smoother mesh, as indicated by the more continuous zebra strips, but requires a larger transition region with more distinct shapes.

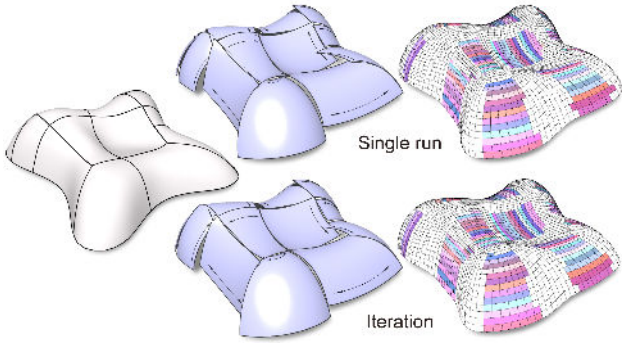


Fig. 16. By iteratively setting the smoothed geometry after post-processing as the target surface and performing additional optimization, one could further reduce the number of distinct elements, while keeping the overall geometry nearly unchanged.

properties that facilitate beam placement [Liu et al. 2006b; Pottmann and Wallner 2008].

$$E_p = \sum_{i=1}^{n_P} \sum_{i=1}^{n_b-1} (||V_i||^2) \tag{12}$$

Controllable panel dimension. Panel dimensions, linked to mesh resolution, can be controlled by adjusting the target edge length (l^*) during tessellation, as shown in Figure 19. Generally, a denser mesh reduces the ratio of distinct shapes to all faces, exhibiting increased cost-effectiveness.

Paneling. For the actual paneling (i.e., designing the panel shape) of the obtained quad mesh, we propose two strategies, as shown in Figure 21. The first option is to use bilinear surfaces, allowing for straight beams that are easy to fabricate, resulting in a faceted appearance. Alternatively, faces within fixed regions can be paneled based on corresponding rotational patches, with transition regions paneled using surfaces with minimal curvature variation, derived through third-order Laplacian smoothing on a high-resolution mesh. This yields a smoother visual appearance but requires non-planar curved beams, adding complexity to the manufacturing process.

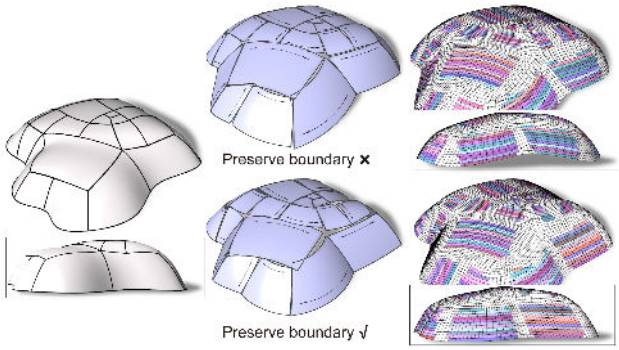


Fig. 17. Our algorithm can preserve the boundaries of the input surface.

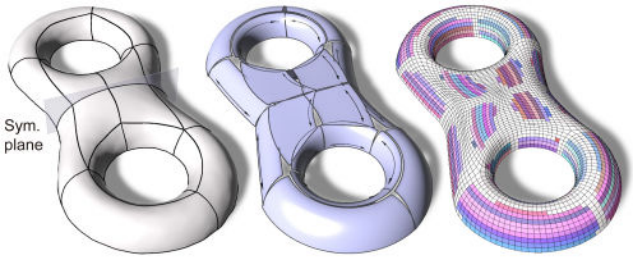


Fig. 18. Our method can maintain the symmetry of the overall geometry.

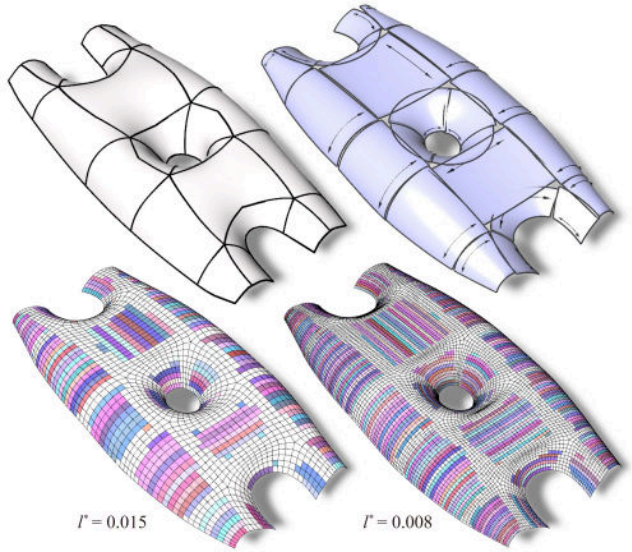


Fig. 19. Using a smaller target edge length results in a denser mesh, which typically allows a larger portion of element shapes to be repeated.

4.2 Special Surfaces Composed of C^1-continuous Patches

In general, our results depend on the input surface, with surfaces that inherently contain rotational features allowing for a greater portion of elements to be repeated. Here, we highlight a special family of surfaces composed of C^1-continuous rotational patches, as shown in Figure 22. These surfaces are created by rotating a profile curve along a series of tangent arcs. The arc spline can be

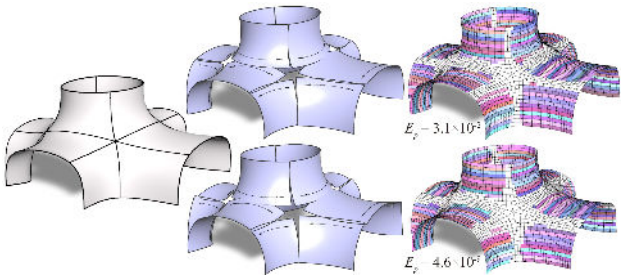


Fig. 20. By requiring the profile curves to be coplanar with the rotational axis, we achieve faces with improved planarity.

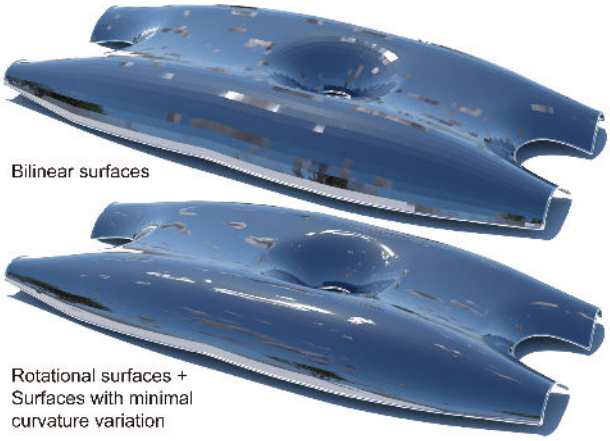


Fig. 21. We propose two strategies for the realistic paneling of the resulting quad mesh. (Top) Paneling with bilinear surfaces, leading to a faceted appearance. (Bottom) Paneling using rotational surfaces for fixed regions and surfaces with minimal curvature variation for transition regions, resulting in a smoother visual appearance.

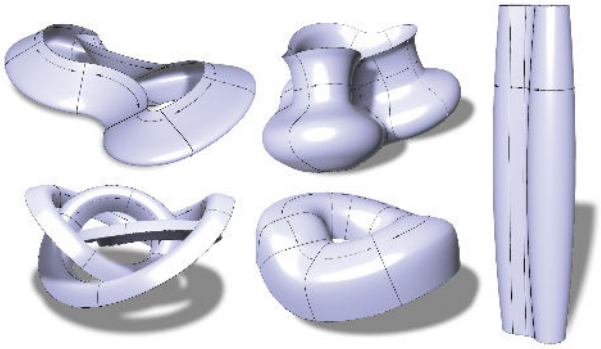


Fig. 22. A special family of surfaces composed of C^1-continuous rotational patches, created by sweeping a profile curve along a series of tangent arcs.

derived by approximating an arbitrary input curve using various methods, such as the one described in Mizutani and Kawaguchi [2021]. For closed curves, the twist must be nulled to align the start and end curve frames, as described in Wang et al. [2008]. These surfaces are inherently smooth, with no gaps between patches and no need for transition regions, making them highly desirable for architectural applications. However, the available design space for such surfaces remains relatively limited.

Table 1. Statistics and Timings

Fig	n_p	$n_f/k_f/n_t/r$	$E_c/E_g/E_n$	t(min)
1	22	3,537/223/1296/0.571	0.279/0.024/0.031	0.67
13(t)	11	3,434/168/874/0.697	1.086/0.009/0.055	2.31
13(b)	11	3,744/194/1368/0.583	0.161/0.131/1.874	2.33
14(t)	2	2,916/72/573/0.779	0.371/0.047/8.524	2.51
14(m)	6	3,936/143/1395/0.609	0.117/0.116/6.091	2.78
14(b)	17	4,354/193/2476/0.387	0.180/0.304/14.679	2.99
16(t)	14	3,171/157/1829/0.374	0.170/0.219/5.294	3.53
16(b)	14	3,171/161/1729/0.403	0.108/0.184/3.007	4.12
17(t)	26	6,108/380/2588/0.514	0.345/0.028/0.917	1.49
17(b)	26	6,527/394/2946/0.488	1.033/0.075/7.400	1.34
18	20	3,512/73/735/0.770	0.220/0.036/0.486	1.12
19(l)	22	2,179/211/971/0.458	0.266/0.030/0.056	0.49
19(r)	22	7,690/394/2490/0.625	0.266/0.030/0.056	0.61
20(t)	16	4,176/206/1261/0.648	0.368/0.048/0.833	0.35
20(b)	16	4,143/197/1428/0.608	0.664/0.068/1.474	0.38
23	23	2,769/191/1441/0.411	0.111/0.111/0.432	3.40

We report each example with the number of patches (n_p); the number of faces (n_f), layers along the arc direction (k_f), transition regions faces (n_t), and the reduction ratio of distinct shapes $r = 1 - (k_f + n_t)/n_f$; the errors for closeness (E_c), gap closure (E_g), surface smoothness (E_n); and the total runtime t.

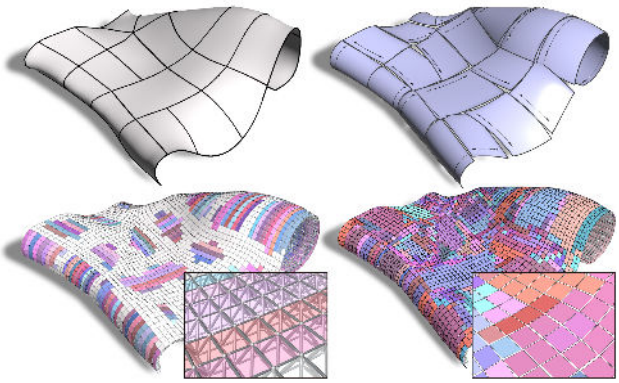


Fig. 23. In applications where gaps are allowed between adjacent faces, our design can be taken as input to further reduce the number of distinct faces using a clustering-based approach [Liu et al. 2024]. Nodes and beams are unmodified, maintaining the repetition derived from the original design.

4.3 Integration with Clustering-based Approach

In specific applications, panels are installed in a separate layer, attached to the underlying beam-node structural layer via connectors, allowing for gaps between adjacent faces (for a practical example, see [Sidelko 2013]). This configuration provides the flexibility to further reduce the number of distinct faces, as shown in Figure 23. Initially, our design consists of 2,769 faces, including 1,632 types of distinct shapes, with all elements seamlessly connected. Using the base mesh as input, we apply a clustering-based approach [Liu et al. 2024] to achieve congruent faces across different rows and patches, and merge transition faces into existing groups.

This further reduces the number of distinct faces to 100, where gaps are introduced in between (with a similarity error of 0.049, measured using their definition). Nodes and beams are unmodified, maintaining the repetition derived from the original design.

5 Conclusion

We present the first method to approximate a given free-form surface using an assembly of untrimmed rotational patches, enabling the simultaneous repetition of multiple element types for cost-effective fabrication. We introduce two similarity metrics to quantify the resemblance of a mesh to a rotational patch. Our segmentation approach divides a surface into quad patches that resemble rotational patches within a prescribed threshold. Our B-spline-based optimization framework can refine rotational patches for faithful surface approximation and smooth connections, while incorporating additional constraints. To meet practical needs, we provide a post-processing tool that converts the discrete patch assembly into a seamless, smooth quad mesh, composed of locally repeated elements. We show that our method is applicable to a variety of free-form surfaces, and capable of addressing diverse architectural requirements.

Practical applications. Our approach offers several unique advantages for practical applications. First, it ensures exactly congruent elements within each group, making it well suited for mold-based fabrication techniques, such as casting and vacuum forming, where each unique shape requires a customized mold. Second, unlike most existing methods that focus on a single element type, our approach simultaneously optimizes multiple element types, even across layers in multi-layer space frames. While this stricter constraint may lead to a lower reduction rate in the number of unique shapes, achieving a balanced reduction across all elements can potentially be more cost-effective than achieving a high reduction in a single type while leaving others unoptimized. Third, our method ensures seamless connections between components, which is critical for high-precision applications such as node assembly, where beams must fit precisely to avoid inducing undesirable internal stresses. Our approach can also be adapted for scenarios with permissive tolerances, such as allowing small gaps between adjacent panels. As shown in Figure 23, we further apply global clustering and optimization to panels, achieving an increased reduction rate of 96.4% (within a prescribed error threshold), while maintaining seamless connectivity and original rotational symmetry for nodes.

Limitations and future work. Our work has several limitations that open up interesting directions for future research. First, transition regions are required to address residual errors, which introduce additional distinct shapes. Minimizing these regions would be a key research direction. Two potential strategies include (1) identifying surface types that can be well approximated using rotational patches and (2) exploring alternative surface segmentation strategies to enhance the outcome. It is also interesting to integrate interactive sketching tools into the patch layout design process to facilitate real-time user-driven surface segmentation, improving design flexibility. In addition, our approach currently focuses on rotational patches; extending it to more general surface types, such as translational, helical, and spherical surfaces, could expand design freedom while ensuring cost-effective element fabrication.

Acknowledgments

We thank the reviewers for their valuable comments.

References

Guy Austern, Isaac Guedi Capeluto, and Yasha Jacob Grobman. 2018. Rationalization methods in computer aided fabrication: A critical review. *Automation in Construction* 90 (June 2018), pp. 281–293. DOI: https://doi.org/10.1016/j.autcon.2017.12.027

Claus Bendtsen. 1996. FADBAD, a flexible C++ package for automatic differentiation - using the forward and backward method. Retrieved from https://api.semanticscholar.org/CorpusID:53910006

Minghao Bi, Yuanpeng Liu, Tao Xu, Yunzhen He, Jiaming Ma, Zicheng Zhuang, and Yi Min Xie. 2024. Clustering and optimization of nodes, beams and panels for cost-effective fabrication of free-form surfaces. *Engineering Structures* 307 (2024), 117912. DOI: https://doi.org/10.1016/j.engstruct.2024.117912

Pengbo Bo, Helmut Pottmann, Martin Kilian, Wenping Wang, and Johannes Wallner. 2011. Circular arc structures. *ACM Transactions on Graphics* 30, 4, Article 101 (jul 2011), 12 pages. DOI: https://doi.org/10.1145/2010324.1964996

Alexander I. Bobenko and Emanuel Huhnen-Venedey. 2012. Curvature line parametrized surfaces and orthogonal coordinate systems: Discretization with dupin cyclides. *Geometriae Dedicata* 159 (August 2012), 207–237. DOI: https://doi.org/10.1007/s10711-011-9653-5

Sergey Bochkanov. ALGLIB. Retrieved from https://www.alglib.net

David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013. Quad-mesh generation and processing: A survey. *Computer Graphics Forum* 32, 6 (2013), 51–76. DOI: https://doi.org/10.1111/cgf.12014

Mario Botsch and Leif Kobbelt. 2004. An intuitive framework for real-time freeform modeling. In *Proceedings of the ACM SIGGRAPH 2004 Papers (SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 630–634. https://doi.org/10.1145/1186562.1015772

Jan Brütting, Gennaro Senatore, and Corentin Fivet. 2021. Design and fabrication of a reusable kit of parts for diverse structures. *Automation in Construction* 125 (May 2021), pp. 103614. DOI: https://doi.org/10.1016/j.autcon.2021.103614

Marcel Campen. 2017. Partitioning surfaces into quadrilateral patches: A survey. *Computer Graphics Forum* 36, 8 (2017), 567–588. DOI: https://doi.org/10.1111/cgf.13153

Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Dual loops meshing: Quality quad layouts on manifolds. *ACM Transactions on Graphics* 31, 4, Article 110 (July 2012), 11 pages. DOI: https://doi.org/10.1145/2185520.2185606

Desai Chen, Pitchaya Sitthi-amorn, Justin T. Lan, and Wojciech Matusik. 2013. Computing and fabricating multiplanar models. *Computer Graphics Forum* 32, 2pt3 (2013), 305–315. DOI: https://doi.org/10.1111/cgf.12050

Rulin Chen, Pengyun Qiu, Peng Song, Bailin Deng, Ziqi Wang, and Ying He. 2023. Masonry shell structures with discrete equivalence classes. *ACM Transactions on Graphics* 42, 4, Article 115 (July 2023), 12 pages. DOI: https://doi.org/10.1145/3592095

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. *ACM Transactions on Graphics* 23, 3 (Aug 2004), 905–914. DOI: https://doi.org/10.1145/1015706.1015817

Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Designing n-polyvector fields with complex polynomials. *Computer Graphics Forum* 33, 5 (Aug 2014), 1–11.

Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J. Mitra, Helmut Pottmann, and Mark Pauly. 2010. Paneling architectural freeform surfaces. *ACM Transactions on Graphics* 29, 4, Article 45 (Jul. 2010), 10 pages. DOI: https://doi.org/10.1145/1778765.1778782

Chi-Wing Fu, Chi-Fu Lai, Ying He, and Daniel Cohen-Or. 2010. K-set tilable surfaces. *ACM Transactions on Graphics* 29, 4, Article 44 (July 2010), 6 pages. DOI: https://doi.org/10.1145/1778765.1778781

Alec Jacobson and Daniele Panozzo 2018. libigl: A simple C++ geometry processing library. (2018). Retrieved from https://libigl.github.io/.

Elias Jadon, Bernhard Thomaszewski, Aleksandra Anna Apolinarska, and Roi Poranne. 2022. Continuous deformation based panelization for design rationalization. In *Proceedings of the SIGGRAPH Asia 2022 Conference Papers (SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 44, 8 pages. DOI: https://doi.org/10.1145/3550469.3555414

Antiopi Koronaki, Paul Shepherd, and Mark Evernden. 2020. Rationalization of freeform space-frame structures: Reducing variability in the joints. *International Journal of Architectural Computing* 18, 1 (2020), pp. 84–99. DOI: https://doi.org/10.1177/14780771198948

Ting-Uei Lee, Yuanpeng Liu, and Yi Min Xie. 2022. Dividing a sphere hierarchically into a large number of spherical pentagons using equal area or equal length optimization. *Computer-Aided Design* 148 (July 2022), pp. 103259. DOI: https://doi.org/10.1016/j.cad.2022.103259

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics* 21, 3 (July 2002), 362–371. DOI: https://doi.org/10.1145/566654.566590

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing (SGP '08)*. Eurographics Association, Goslar, DEU, 1495–1504.

Yuanpeng Liu, Ting-Uei Lee, Antiopi Koronaki, Nico Pietroni, and Yi Min Xie. 2023. Reducing the number of different nodes in space frame structures through clustering and optimization. *Engineering Structures* 284 (2023), 116016. DOI: https://doi.org/10.1016/j.engstruct.2023.116016

Yuanpeng Liu, Ting-Uei Lee, Anooshe Rezaee Javan, Nico Pietroni, and Yi Min Xie. 2024. Reducing the number of different faces in free-form surface approximations through clustering and optimization. *Computer-Aided Design* 166 (2024), 103633. DOI: https://doi.org/10.1016/j.cad.2023.103633

Yuanpeng Liu, Ting-Uei Lee, Anooshe Rezaee Javan, and Yi Min Xie. 2022. Extending goldberg's method to parametrize and control the geometry of goldberg polyhedra. *Royal Society Open Science* 9, 8 (2022), pp. 220675. DOI: https://doi.org/10.1098/rsos.220675

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006b. Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics* 25, 3 (July 2006), 681–689. DOI: https://doi.org/10.1145/1141911.1141941

Yang Liu, Helmut Pottmann, and Wenping Wang. 2006a. Constrained 3D shape reconstruction using a combination of surface fitting and registration. *Comput. Aided Des.* 38, 6 (2006), 572–583. DOI: https://doi.org/10.1016/j.cad.2006.01.014

Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Transactions on Graphics* 30, 6 (Dec 2011), 1–10. DOI: https://doi.org/10.1145/2070781.2024174

Zhongyuan Liu, Zhan Zhang, Di Zhang, Chunyang Ye, Ligang Liu, and Xiao-Ming Fu. 2021. Modeling and fabrication with specified discrete equivalence classes. *ACM Transactions on Graphics* 40, 4, Article 41 (July 2021), 12 pages. DOI: https://doi.org/10.1145/3450626.3459843

Marco Livesu, Nico Pietroni, Enrico Puppo, Alla Sheffer, and Paolo Cignoni. 2020. LoopyCuts: Practical feature-preserving block decomposition for strongly hex-dominant meshing. *ACM Transactions on Graphics* 39, 4, Article 121 (Aug 2020), 17 pages. DOI: https://doi.org/10.1145/3386569.3392472

Hongjia Lu and Yi Min Xie. 2023. Reducing the number of different members in truss layout optimization. *Structural and Multidisciplinary Optimization* 66, 3 (2023), 52. DOI: https://doi.org/10.1007/s00158-023-03514-y

Allen R. Martin. 1982. *Principal Patches for Computational Geometry*. Dissertation. University of Cambridge. Retrieved from https://api.semanticscholar.org/CorpusID:117700063

Keisuke Mizutani and Ken'ichi Kawaguchi. 2021. Curve approximation by G1 arc splines with a limited number of types of curvature and length. *Computer Aided Geometric Design* 90 (2021), 102036. DOI: https://doi.org/10.1016/j.cagd.2021.102036

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Transactions on Graphics* 33, 4, Article 135 (July 2014), 14 pages. DOI: https://doi.org/10.1145/2601097.2601154

Stefano Nuvoli, Alex Hernandez, Claudio Esperança, Riccardo Scateni, Paolo Cignoni, and Nico Pietroni. 2019. QuadMixer: Layout preserving blending of quadrilateral meshes. *ACM Transactions on Graphics* 38, 6, Article 180 (nov 2019), 13 pages. DOI: https://doi.org/10.1145/3355089.3356542

Davide Pellis, Martin Kilian, Helmut Pottmann, and Mark Pauly. 2021. Computational design of weingarten surfaces. *ACM Transactions on Graphics* 40, 4, Article 114 (July 2021), 11 pages. DOI: https://doi.org/10.1145/3450626.3459939

Nico Pietroni, Corentin Dumery, Raphael Falque, Mark Liu, Teresa Vidal-Calleja, and Olga Sorkine-Hornung. 2022. Computational pattern making from 3D garment models. *ACM Transactions on Graphics* 41, 4, Article 157 (July 2022), 14 pages. DOI: https://doi.org/10.1145/3528223.3530145

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable feature-line driven quad-remeshing. *ACM Transactions on Graphics* 40, 4, Article 155 (July 2021), 17 pages. DOI: https://doi.org/10.1145/3450626.3459941

Nico Pietroni, Enrico Puppo, Giorgio Marcias, Roberto Scopigno, and Paolo Cignoni. 2016. Tracing field-coherent quad layouts. *Computer Graphics Forum* 35, 7 (2016), 485–496. DOI: https://doi.org/10.1111/cgf.13045

Kacper Pluta, Michal Edelstein, Amir Vaxman, and Mirela Ben-Chen. 2021. PH-CPF: Planar hexagonal meshing using coordinate power fields. *ACM Transactions on Graphics* 40, 4, Article 156 (July 2021), 19 pages. DOI: https://doi.org/10.1145/3450626.3459770

Helmut Pottmann, Michael Eigensatz, Amir Vaxman, and Johannes Wallner. 2015. Architectural geometry. *Computers and Graphics* 47 (April 2015), pp. 145–164. DOI: https://doi.org/10.1016/j.cag.2014.11.002

Helmut Pottmann and Thomas Randrup. 1998. Rotational and helical surface approximation for reverse engineering. *Computing* 60 (1998), 307–322. DOI: https://doi.org/10.1007/BF02684378

Helmut Pottmann and Johannes Wallner. 2001. *Computational Line Geometry*. Vol. 6. Springer.

Helmut Pottmann and Johannes Wallner. 2008. The focal geometry of circular and conical meshes. *Advances in Computational Mathematics* 29 (jul 2008), 249–268. DOI: https://doi.org/10.1007/s10444-007-9045-4

Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. 2015. Perfect matching quad layouts for manifold meshes. *Computer Graphics Forum* 34, 5 (2015), 219–228. DOI: https://doi.org/10.1111/cgf.12710

Jason Sidelko. 2013. *Museo Soumaya: Facade Design to Fabrication*. Lulu. com.

Patricio D. Simari and Karan Singh. 2005. Extraction and remeshing of ellipsoidal representations from mesh data. In *Proceedings of Graphics Interface 2005 (GI '05)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 161–168.

Mayank Singh and Scott Schaefer. 2010. Triangle surfaces with discrete equivalence classes. *ACM Transactions on Graphics* 29, 4, Article 46 (July 2010), 7 pages. DOI: https://doi.org/10.1145/1778765.1778783

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of triangle meshes. *ACM Transactions on Graphics* 37, 4, Article 77 (jul 2018), 14 pages. DOI: https://doi.org/10.1145/3197517.3201303

Amir Vaxman, Marcel Campen, Olga Diamanti, David Bommes, Klaus Hildebrandt, Mirela Ben-Chen Technion, and Daniele Panozzo. 2017. Directional field synthesis, design, and processing. In *Proceedings of the ACM SIGGRAPH 2017 Courses (SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 12, 30 pages. DOI: https://doi.org/10.1145/3084873.3084921

Wenping Wang, Bert Jüttler, Dayue Zheng, and Yang Liu. 2008. Computation of rotation minimizing frames. *ACM Transactions on Graphics* 27, 1, Article 2 (mar 2008), 18 pages. DOI: https://doi.org/10.1145/1330511.1330513

Wenping Wang, Helmut Pottmann, and Yang Liu. 2006. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics* 25, 2 (apr 2006), 214–238. DOI: https://doi.org/10.1145/1138450.1138453

Jianhua Wu and Leif Kobbelt. 2005. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum* 24, 3 (2005), 277–284. DOI: https://doi.org/10.1111/j.1467-8659.2005.00852.x

Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. 2012. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design* 44, 11 (2012), 1072–1082. DOI: https://doi.org/10.1016/j.cad.2012.04.005

Zheng-Yu Zhao, Mo Li, Zheng Zhang, Qing Fang, Ligang Liu, and Xiao-Ming Fu. 2023. Evolutionary piecewise developable approximations. *ACM Transactions on Graphics* 42, 4, Article 120 (July 2023), 14 pages. DOI: https://doi.org/10.1145/3592140

Tianyu Zhu, Zeng-Hao Xu, Ligang Liu, and Xiao-Ming Fu. 2023. Modeling with discrete equivalence classes of planar quads. *Computers and Graphics* 115 (2023), 404–411. DOI: https://doi.org/10.1016/j.cag.2023.07.032

Henrik Zimmer, Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Rationalization of triangle-based point-folding structures. *Computer Graphics Forum* 31, 2pt3 (2012), 611–620. DOI: https://doi.org/10.1111/j.1467-8659.2012.03040.x

Henrik Zimmer and Leif Kobbelt. 2014. Zometool rationalization of freeform surfaces. *IEEE Transactions on Visualization and Computer Graphics* 20, 10 (2014), 1461–1473. DOI: https://doi.org/10.1109/TVCG.2014.2307885

Henrik Zimmer, Florent Lafarge, Pierre Alliez, and Leif Kobbelt. 2014. Zometool shape approximation. *Graphical Models* 76, 5 (September 2014), pp. 390–401. DOI: https://doi.org/10.1016/j.gmod.2014.03.009

Appendix

A Curvature-based Distance Term

In our implementation, we employ a curvature-based distance term for B-spline curve fitting, as introduced in [Wang et al. 2006] for enhanced efficiency. The formulation is detailed in Equation (13), where d_k denotes the signed distance from the data point X_k to the nearest point $P(t_k)$ on the B-spline curve, with t_k being the corresponding parametric value. The sign of d_k depends on the relative positions of X_k and the curvature center C_k at $P(t_k)$; it is negative if they are on opposite sides of the curve, and positive otherwise. Additional parameters include the radius of curvature ρ_k, the tangent vector T_k, and the normal vector N_{Bk} at point $P(t_k)$. The formulation $P(t) = \sum_{i=1}^{m} P_i B_i(t)$ calculates the point on the B-spline curve using m control points P_i and m basis functions $B_i(t)$, related to the curve's knot vector and order. For more details, we refer to [Wang et al. 2006].

$$e_{dk} = \begin{cases} \frac{d_k}{d_k - \rho_k} \left((P(t_k) - X_k)^T T_k \right)^2 \\ \quad + \left((P(t_k) - X_k)^T N_{Bk} \right)^2, & \text{if } d_k < 0, \\ \left((P(t_k) - X_k)^T N'_k \right)^2, & \text{if } 0 \le d_k < \rho_k, \end{cases} \tag{13}$$

Received 4 October 2024; revised 27 March 2025; accepted 3 June 2025