Efficient Piecewise Higher-Order Parametrization of Discrete Surfaces with Local and Global Injectivity

Manish Mandad, Marcel Campen

Osnabrück University, Germany

Abstract

The parametrization of triangle meshes, in particular by means of computing a map onto the plane, is a key operation in computer graphics. Typically, a piecewise linear setting is assumed, i.e., the map is linear per triangle. We present a method for the efficient computation and optimization of piecewise nonlinear parametrizations, with higher-order polynomial maps per triangle. We describe how recent advances in piecewise linear parametrization, in particular efficient second-order optimization based on majorization, as well as practically important constraints, such as local injectivity, global injectivity, and seamlessness, can be generalized to this higher-order regime. Not surprisingly, parametrizations of higher quality, i.e., lower distortion, can be obtained that way, as we demonstrate on a variety of examples.

Keywords: Bézier triangles, curved meshes, Hessian majorization.

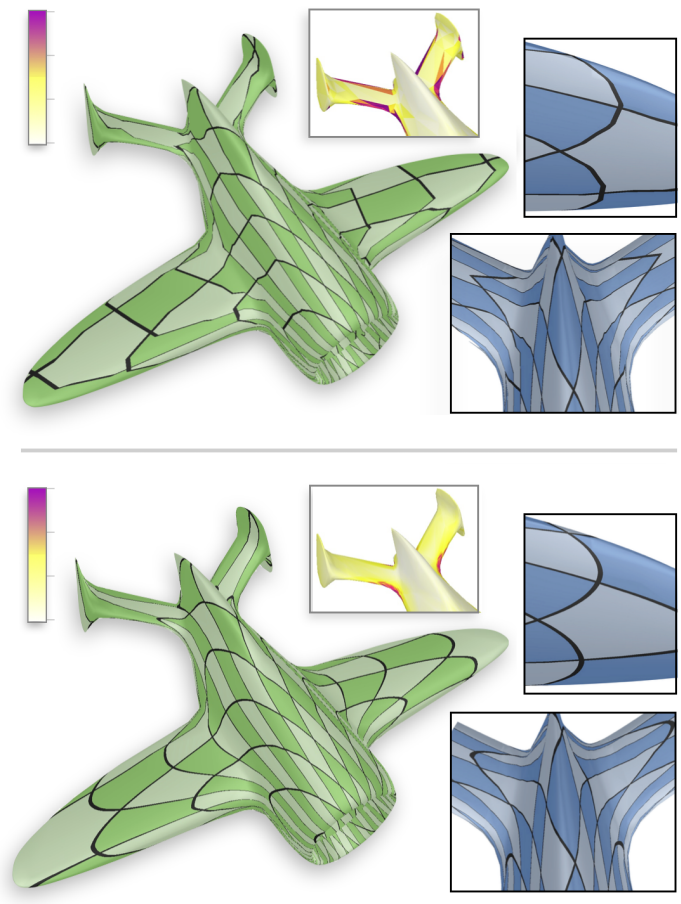


Figure 1: Top: standard piecewise linear parametrization of a triangle mesh. Bottom: piecewise cubic parametrization, computed using our method. Both have been optimized for low distortion in terms of the symmetric Dirichlet energy. Using a higher-order parametrization, lower distortion (here by 16%, also see the heat map insets, visualizing the pointwise mapping distortion) as well as visually higher quality is achieved. The visual difference is particularly clear here because the underlying triangle mesh is quite coarse (1.5K triangles), but also for finer versions the effect can be observed (blue insets on the right, 11.2K triangles).

1. Introduction

An essential operation in computer graphics and geometry processing applications is the parametrization of discrete surfaces, in particular by means of mapping a triangle mesh onto a planar domain. Prominent use cases are remeshing and texturing. Such parametrizations are most commonly computed, represented, and processed in a piecewise linear manner, with a linear (or affine) map per triangle. By contrast, we describe a method to efficiently generate and optimize low-distortion parametrizations using piecewise nonlinear, higher-order mapping functions.

The increased flexibility provided by this higher-order polynomial representation enables parametrizations of lower distortion and higher visual quality, as demonstrated in Fig. 1. The use of higher-order basis functions on unstructured simplicial meshes has been investigated for a long time [1], and advantages have been demonstrated in various contexts such as simulation [2] or deformation [3] before. We focus here for the first time on the question how and to what extent the problem of injective surface parametrization can benefit in terms of reduced parametric distortion – a key aspect in this domain. This will help estimating whether the benefit is worth the added cost in concrete use cases.

Our method has relations to and shares some technical aspects with algorithms for the generation or optimization of curved planar meshes, which are of interest in the field of finite element computations. At the same time, it has novel features and details, which are of particular relevance in the parametrization context – but may benefit the field of curved mesh generation as well.

Concretely, in this paper we concisely describe how to:

- practically evaluate popular distortion objectives in the higher-order setting – which, due to the Jacobian not being constant per triangle, requires additional effort;
- apply the concept of composite majorization [4] in this setting, enabling efficient second-order optimization using the distortion objective's gradient and Hessian;
- ensure local injectivity of the parametrization – which in the

nonlinear setting is a per-point, not a per-triangle property – using an improved injectivity condition evaluation;

- ensure, optionally, global injectivity (or bijectivity) by means of a nonlinear version of an ambient space triangulation [5, 6];
- compute global higher-order parametrizations with cone singularities and seamless transitions, as these are important for purposes such as quadrilateral remeshing;
- modify meshes by means of edge flips for purposes of mesh optimization while preserving the validity of an underlying higher-order parametrization.

2. Related Work

Many of the underlying techniques we make use of have been proposed and employed before in other contexts, e.g., in piecewise linear parametrization or in higher-order mesh generation, as reviewed below. We adapt, improve, specialize, or generalize these to our problem setting, suitably combine them to form an efficient method, and spell out the necessary details specific to the parametrization problem in a self-contained manner.

2.1. Linear Parametrization

Being an important tool in many areas of computer graphics, the literature on computing, optimizing, and using piecewise linear parametrizations of triangle meshes is vast [7, 8]. Recent results in this field concern advanced aspects, like support for singularities and cuts to enable global parametrization of topologically non-trivial surfaces [9, 10, 11, 12, 13, 14, 15, 16], control over local or global injectivity [17, 18, 19, 5], and means to speed-up the optimization, using regularizers, preconditioners, proxies, etc. [20, 21, 22, 23, 4, 24]. We adopt some of the ideas developed in this linear setting and apply them in the higher-order case.

2.2. Nonlinear Parametrization

The use of piecewise nonlinear mapping for triangle meshes is less common, in particular when it comes to global parametrization. In the context of procedural texturing, nonlinear functions are sometimes employed locally, e.g. in [25]. In the context of subdivision, [26] mention the application of subdivision schemes to texture coordinates, effectively leading to nonlinear texture mapping functions. Later, generation and optimization of subdivision-based maps was considered [27, 28]. It remains an open question how aspects of injectivity or bijectivity could be addressed in this subdivision context.

2.3. Curved Mesh Generation

The image of a (straight-edge) triangle mesh under a nonlinear parametrization is a planar mesh with curved edges in parameter space. Planar meshes with curved edges have been of interest for simulation purposes since their proposal in the 1960s [1]. The most common approach for the generation of such curved meshes is the a posteriori curving of initially generated linear, straight-edge meshes, as discussed in [29]. This is related to the problem at hand because the resulting deformation is a higher-order map, effectively parameterizing the initial mesh in a specific way.

A large number of methods for this purpose of triangular (or tetrahedral) mesh curving have been described. Some similarity with our method in terms of a related objective is shared by those that perform the deformation based on some measure of distortion [30, 31, 32, 33, 34, 35, 36]. Others are driven by physical analoga, e.g., elasticity models [37, 38, 39, 40, 41, 42, 43]. Close ties of these to the above distortion minimization techniques are discussed by [44, 45].

The main goal in this field of mesh curving is to adapt to some smooth model boundary [46, 47]. The required deformation is relatively small in this context, with displacements commonly of the same order of magnitude as the sizes of the triangle elements. Only for highly anisotropic meshes larger deformations can be necessary [48]. In our parametrization context, required displacements can be orders of magnitude larger, putting significantly higher efficiency requirements on the optimization, cf. Sec. 4.

A further difference lies in these methods typically not guaranteeing local injectivity of the result. While the underlying energy or model is commonly designed to promote injectivity, due to discretization and due to local minima, guarantees commonly cannot be given. Modification of the mesh (if permissible) can increase the probability of success [49, 50, 47, 46, 29]. By contrast, we start from a locally injective initial parametrization, and are able to optimize for low distortion while strictly preserving this property. This is due to the difference that our goal is not to ultimately conform to a specific model boundary, but to minimize the distortion, as far as possible within the subspace of injective parametrizations.

A final small difference is the relevant reference for distortion: while mesh curving techniques target minimal distortion relative to the initial linear mesh, we need to optimize relative to a given surface mesh, not its initial image in the parameter plane.

2.4. Injectivity

Local Injectivity. Questions of injectivity have been of interest for higher-order maps of triangles since the early days of nonlinear finite element analysis [51]. In contrast to linear maps, where the question of local injectivity (i.e., positivity of the map Jacobian's determinant) reduces to a simple orientation check of three points, closed-form expressions to answer this for higher-order maps are not available. Various sufficient or necessary conditions have been proposed and employed [52, 29, 53, 54, 36, 55, 56]. We discuss these in more detail in Sec. 5.

Many of the above curved mesh generation techniques strive (explicitly or implicitly) to repair non-injective configurations, but providing guarantees of success is an extremely involved endeavor. When an injective initial configuration is available, an alternative approach is to strictly remain in an injective state during optimization of the parametrization. To this end commonly barrier terms are included (e.g. inverse spline [57] or log barriers [58, 30]) and explicit injectivity violation checks are performed during a line search for valid update steps [57, 18, 19]. In our method, we combine higher-order injectivity tests with this principle of explicit violation prevention, cf. Sec. 5.

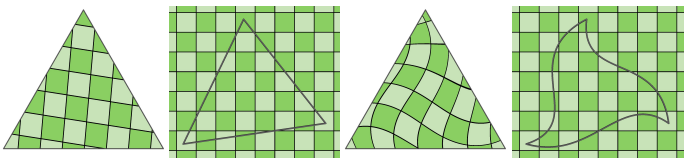


Figure 2: Illustration of a linear (left) and a cubic (right) exemplary parametrization of a single triangle, visualized using a mapped texture.

Global Injectivity. When a strict one-to-one mapping is required, e.g. for texturing applications, global injectivity of the parametrization is relevant. When starting from a globally bijective initial state, a violation of this property during optimization is necessarily preceded by a collision of the image's boundary in parameter space. Hence, classical collision detection and prevention techniques can be employed to preserve global injectivity [59, 60].

Spatial partitioning data structures are commonly employed in this context to speedup interference queries. An instance of this is a triangulation of the ambient space surrounding an object [61], in our case the mesh's image in parameter space [5, 6]. By deforming this triangulation with the mesh's image during optimization, interferences imply flipped triangles in this triangulation, making detection easy.

This technique cannot easily be applied in our higher-order setting, as the ambient triangulation problem turns into an involved constrained curved meshing problem. We instead employ a local remeshing strategy for the higher-order ambient mesh. Related curved mesh modifications have been employed, e.g., by [50] and [62]; we employ improved variants that take injectivity of the map into account, cf. Sec. 6.

3. Higher-Order Map

Let t be a triangle in $\mathbb{R}^3$ with barycentric coordinates $\xi = (\xi_0, \xi_1, 1-\xi_0-\xi_1)$; we will sometimes use ξ_2 as a shorthand for $1-\xi_0-\xi_1$. A map $f : t \to \mathbb{R}^2$ of polynomial degree n over this domain in Bernstein-Bézier form [63] is defined as

$$f(\xi) = \begin{pmatrix} u(\xi) \\ v(\xi) \end{pmatrix} = \sum_{i+j+k=n} \begin{pmatrix} u_{ijk} \\ v_{ijk} \end{pmatrix} B^n_{ijk}(\xi), \tag{1}$$

with $n(n+1)/2$ coefficients (*control points*) $c_{ijk} = (u_{ijk}, v_{ijk}) \in \mathbb{R}^2$ and triangular Bernstein basis functions

$$B^n_{ijk}(\xi) = \frac{n!}{i!\,j!\,k!} \xi_0^i \xi_1^j \xi_2^k. \tag{2}$$

In Fig. 2 such triangular maps are illustrated for the linear case, $n = 1$, and the cubic case, $n = 3$. The main objective of the method presented herein is the determination of coefficients c_{ijk} for each triangle of a mesh such that the resulting combined map has the desired properties, such as low parametric distortion and injectivity.

3.1. Derivatives

To obtain the Jacobian of f, it is convenient to consider the map f as being composed of two maps, $f = \phi \circ \psi^{-1}$, via a right triangle with unit length legs, as illustrated in Fig. 3. The map $\psi : T \to t$ is a simple affine map, while $\phi : T \to \mathbb{R}^2$ is a Bézier map of degree n.

The Jacobian of the Bézier map ϕ is easily built from the partial derivatives by ξ_0 and ξ_1:

$$J_\phi(\xi) = \begin{bmatrix} \frac{\partial u}{\partial \xi_0}(\xi) & \frac{\partial u}{\partial \xi_1}(\xi) \\ \frac{\partial v}{\partial \xi_0}(\xi) & \frac{\partial v}{\partial \xi_1}(\xi) \end{bmatrix}, \tag{3}$$

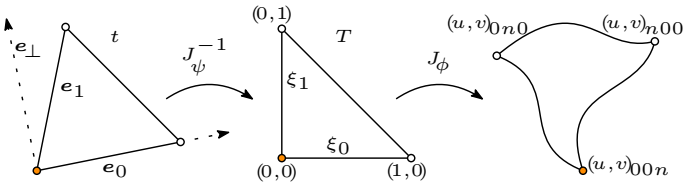


Figure 3: The Jacobian J_f of the mapping function f (from triangle t into parameter space $\mathbb{R}^2$) can be computed as a composition of two Jacobians via intermediate right unit leg triangle T.

where the partial derivatives [63] are given by

$$\begin{aligned} \frac{\partial u}{\partial \xi_0}(\xi) &= n \sum_{i+j+k=n-1} (u_{(i+1)jk} - u_{ij(k+1)}) B^{n-1}_{ijk}(\xi), \\ \frac{\partial u}{\partial \xi_1}(\xi) &= n \sum_{i+j+k=n-1} (u_{i(j+1)k} - u_{ij(k+1)}) B^{n-1}_{ijk}(\xi). \end{aligned} \tag{4}$$

The (constant) Jacobian of the affine map $\psi : T \to t$ is easily computed from the edge vectors $\boldsymbol{e}_0, \boldsymbol{e}_1 \in \mathbb{R}^3$ of triangle t as

$$J_\psi = \begin{bmatrix} \boldsymbol{e}_0^\mathsf{T} \hat{\boldsymbol{e}}_0 & \boldsymbol{e}_1^\mathsf{T} \hat{\boldsymbol{e}}_0 \\ 0 & \boldsymbol{e}_1^\mathsf{T} \hat{\boldsymbol{e}}_\perp \end{bmatrix}, \tag{5}$$

where $\boldsymbol{e}_\perp = \boldsymbol{e}_0 \times \boldsymbol{e}_1 \times \boldsymbol{e}_0$, and $\hat{\boldsymbol{e}}$ denotes a unit vector along $\boldsymbol{e}$. The columns of J_ψ are just the two edge vectors relative to a local orthonormal 2D coordinate system, as illustrated using dashed lines in Fig. 3 left. Then

$$J_\psi^{-1} = \begin{bmatrix} j_0 & j_1 \\ 0 & j_2 \end{bmatrix} = \frac{1}{2A_t} \begin{bmatrix} \boldsymbol{e}_1^\mathsf{T} \hat{\boldsymbol{e}}_\perp & -\boldsymbol{e}_1^\mathsf{T} \hat{\boldsymbol{e}}_0 \\ 0 & \boldsymbol{e}_0^\mathsf{T} \hat{\boldsymbol{e}}_0 \end{bmatrix}. \tag{6}$$

The Jacobian of f is then $J_f(\xi) = J_\phi(\xi) J_\psi^{-1}$. Note that J_ψ^{-1} is independent of ξ, constant per triangle due to ψ being affine, and independent of the parametrization (i.e. of the coefficients c_{ijk}).

3.2. Continuity

Given a triangle mesh M, a parametrization of this mesh is defined by a collection of triangular maps, one for each triangle, as described above. For most applications, it is important that these individual maps join continuously across the edges – at least away from (potentially topologically unavoidable) *cuts* or *seams*.

Along an edge a triangular map in Bernstein-Bézier form is exclusively defined by a subset of its control points. For instance, the $(n+1)$ control points c_{0jk} determine the map restricted to edge $\boldsymbol{e}_1$ in the example from Fig. 3. It is thus sufficient that these *edge control points* coincide between adjacent triangles in order to obtain a parametrization of M that is C^0 continuous, as illustrated in

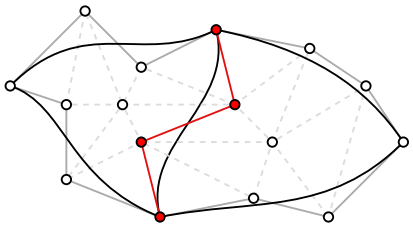


Figure 4: Images of two adjacent triangles under degree 3 maps with their associated control points. The edge control points (red) are shared between the triangles, thus the combined map is C^0.

Fig. 4. In practice, to avoid redundancy, edge control points can be shared among two triangles, and vertex control points (with two zero indices, being edge control points for two edges of a triangle each) shared among all triangles incident to a vertex.

We focus on the C^0 setting herein. For certain applications higher orders of parametrization continuity, i.e., smoothness, may be of interest – but this comes with major additional challenges, cf. Sec. 9.

3.3. Distortion

A variety of measures to quantify a parametrization's metric distortion have been proposed. They can commonly be expressed in terms of the Jacobian's singular values [7]. Particularly in the context of injective parametrization, the symmetric Dirichlet energy [19] has become popular, in part due to its barrier characteristics, naturally preventing degeneracies during distortion-driven optimization. We thus exemplarily focus on this important distortion measure in the following.

Let the two singular values of $J_f(\xi)$ be denoted $\Sigma(\xi), \sigma(\xi)$. Then the symmetric Dirichlet energy at point ξ in triangle t is

$$E^t_{SD}(\xi) = \Sigma(\xi)^2 + \sigma(\xi)^2 + \Sigma(\xi)^{-2} + \sigma(\xi)^{-2}. \tag{7}$$

Integrating this pointwise measure over the entire mesh M yields the total distortion E_{SD} as follows:

$$E^M_{SD} = \sum_{t\in M} \int_t E^t_{SD}(\xi) dxdy. \tag{8}$$

For the usual case of piecewise linear parametrization, $E^t_{SD}(\xi)$ is constant over t. For higher-order cases, where this is not the case, closed-form expressions for the integral are not available. Hence, numerical integration using some quadrature scheme needs to be employed – a standard approach, e.g., in higher-order FEM [64].

Let ξ_l, $l \in Q = \{1, \dots, k\}$, denote the barycentric coordinates of some scheme's k quadrature points, and w_l the associated relative weights. Quadrature then yields

$$E^M_{SD} \approx \bar{E}^M_{SD} = \sum_{t\in M} A_t \sum_{l\in Q} w_l E^t_{SD}(\xi_l) = \sum_{(t,l)\in M\times Q} A_t w_l E^t_{SD}(\xi_l). \tag{9}$$

Note that $E^t_{SD}(\xi_l)$ is not a polynomial but a rational function, such that this quadrature is approximative in general. We discuss the choice of quadrature scheme in Sec. 8.3.

4. Efficient Optimization

Main objective of this paper is to show how nonlinear parametrization maps for triangle meshes can be generated and optimized for low distortion, possibly subject to various constraints. The degrees of freedom are the coefficients c_{ijk} per triangle – which in the classical linear case simply are the three vertices' mapping coordinates (or texture coordinates) per triangle – and we consider $\bar{E}^M_{SD}$ as the distortion to be minimized.

This optimization objective is nonlinear, and we have a potentially very large number of degrees of freedom, $n(n+1)$ per triangle of the mesh M (some shared, cf. Sec. 3.2). For efficiency, we would like to apply Newton's second-order optimization method. It makes use of the objective's Hessian. If this Hessian is not positive semidefinite (PSD), however, the computed update directions are not descent directions in general. The Hessian of $\bar{E}^M_{SD}$ is not PSD (nor that of other popular distortion measures).

4.1. Majorization

In order to obtain a well-justified Hessian proxy that is PSD, we thus adopt in the following the majorization ideas of [4], enabling the efficient optimization of higher-order maps using Newton's method. For clarity we briefly recap the core idea, and spell out the relevant formulas adapted to our setting. One starts by decomposing $E^t_{SD}(\xi) = h \circ g\,(\xi)$, with

$$h\left(\begin{bmatrix}\Sigma\\ \sigma\end{bmatrix}\right) = \Sigma^2 + \Sigma^{-2} + \sigma^2 + \sigma^{-2} \quad \text{and} \quad g(\xi) = \begin{pmatrix}\Sigma(\xi)\\ \sigma(\xi)\end{pmatrix}. \tag{10}$$

Using the following definitions, built from the entries of J_f, cf. Eq. (3) and (6), the singular values of $J_f(\xi)$ appearing in g can be written as $\Sigma(\xi) = \|\alpha(\xi)\| + \|\beta(\xi)\|$ and $\sigma(\xi) = \|\alpha(\xi)\| - \|\beta(\xi)\|$:

$$\alpha = \frac{1}{2}\begin{bmatrix} j_0 \frac{\partial u}{\partial \xi_0} + j_1 \frac{\partial v}{\partial \xi_0} + j_2 \frac{\partial v}{\partial \xi_1} \\ j_0 \frac{\partial v}{\partial \xi_0} - j_1 \frac{\partial u}{\partial \xi_0} - j_2 \frac{\partial u}{\partial \xi_1} \end{bmatrix} \quad \beta = \frac{1}{2}\begin{bmatrix} j_0 \frac{\partial u}{\partial \xi_0} - j_1 \frac{\partial v}{\partial \xi_0} - j_2 \frac{\partial v}{\partial \xi_1} \\ j_0 \frac{\partial v}{\partial \xi_0} + j_1 \frac{\partial u}{\partial \xi_0} + j_2 \frac{\partial u}{\partial \xi_1} \end{bmatrix}$$

Hence, function g can be further decomposed as $g = g^+ + g^-$ with

$$g^+ = \begin{bmatrix}\|\alpha\| + \|\beta\| \\ \|\alpha\|\end{bmatrix}, \quad g^- = \begin{bmatrix}0 \\ -\|\beta\|\end{bmatrix}.$$

This leads to a *convex-concave* decomposition of $E^t_{SD}(\xi)$, where both h and g^+ are convex, while g^- is concave.

The (non-PSD) Hessian of $E^t_{SD}(\xi)$, following the chain rule, is given by

$$H^t_{h\circ g} = \nabla g^\top \nabla^2 h \nabla g + \nabla h^\top \nabla^2 (g^+ + g^-),$$

Note that the first term of this expression is PSD, but the second is not in general. The Hessian of a tight *convex majorizer* can be formulated by discarding the non-PSD components as follows:

- $\nabla^2 g^+$: if multiplied by a negative factor (as g^+ is convex),
- $\nabla^2 g^-$: if multiplied by a positive factor (as g^- is concave).

The Hessian H^t of the convex majorizer of E^t_{SD} is thus given by

$$H^t = \nabla g^\top \nabla^2 h \nabla g \;+\; (\nabla h)^\top_+ \nabla^2 g^+ \;-\; (\nabla h)^\top_- \nabla^2 g^-, \tag{11}$$

where $(\cdot)_+$ and $(\cdot)_-$ are component-wise clamping functions, clamping negative or positive values to zero, respectively.

Due to linearity, recalling Eq. (9), the Hessian H^M of the corresponding convex majorizer of the total energy $\bar{E}^M_{SD}$ is

$$H^M = \sum_{(t,l)\in M\times Q} A_t w_l H^t(\xi_l). \tag{12}$$

All expressions required to compute gradient and Hessian, given control points values, are spelled out explicitly in Appendix A.

4.2. Newton Step and Line Search

Given a parametrization, defined through its control points, let c^0 denote the vector of all these control points' coordinates. Then the update Δc is computed according to Newton's method as

$$\Delta c = -[H^M(c^0)]^{-1} \nabla E^M_{SD}(c^0), \tag{13}$$

where H (Eq. (11)) and the gradient are with respect to the control point coordinates. The update is then applied by setting $c^1 = c^0 + \lambda \Delta c$, where the step size factor λ is determined using a line search which ensures that the objective value (i.e. distortion) decreases and, if desired, injectivity is maintained as described in Sec. 5 ($\det J_f > 0$ and angle sum $\leq 2\pi$ conditions) and Sec. 6.

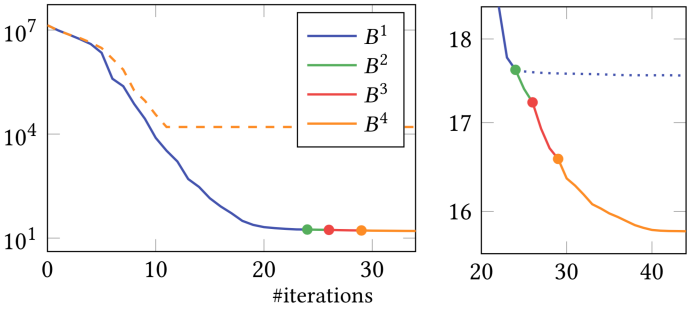


Figure 5: Energy E^M_{SD} (log scale) vs optimization iterations for an example case (model G, $10K$ triangles), for degrees B^1–B^4. When optimizing with high degree from the start, optimization may end early in bad local minima (here for B^4, dashed graph). By instead starting with degree 1 and incrementally elevating the degree (indicated by dots) during optimization (Sec. 4.3), this is avoided and the overall best convergence behavior is achieved (solid graph). On the right a blow-up shows a detailed view of the behavior.

4.3. Incremental Degree Elevation

When starting the optimization from a lower order state – e.g. in our experiments we use a piecewise *linear* Tutte embedding to guarantee injectivity – the optimization process can be sped up and be effectively regularized by incrementally elevating the degree [63]. The degree $n+1$ control points c^{n+1} equivalent to degree n control points c^n are easily computed via

$$c^{n+1}_{ijk} = \frac{1}{n+1}\left(ic^n_{(i-1)jk} + jc^n_{i(j-1)k} + kc^n_{ij(k-1)}\right).$$

As discussed in previous work [62], starting by optimizing a piecewise linear function and then following up with higher-order optimization in particular helps reducing wiggling that higher-order polynomials are prone to, and that may lead to getting stuck early in local minima – an example is illustrated in Fig. 5. We use this simple heuristic: the degree is elevated (until the desired degree is reached) whenever the relative decrease in objective value due to a Newton step drops below 1%.

Remark:. For comparative analyses, it was of interest to be able to successfully perform optimization also without incremental degree elevation. The following proved to be a highly beneficial heuristic for this case (avoiding premature convergence as dashed in Fig. 5): If even after reducing step size λ to below 10^{-5} in the line search there are triangles where local injectivity would be violated, we temporarily exclude them from the objective and restart the line search with the recomputed update. This enabled the successful optimization without incremental elevation for our comparative experiments (in particular Fig. 18 and Fig. 15).

5. Local Injectivity

An important property of parametrizations, required by most applications, is local injectivity. A continuous piecewise map is locally injective iff its Jacobian per piece is non-singular, i.e. $\det J_f \neq 0$, of the same orientation, i.e. orientation preserving ($\det J_f > 0$) or reversing ($\det J_f < 0$), and the sum of unsigned image sector angles is $\leq 2\pi$ around every vertex [65] – though the latter condition is not relevant for all applications. We always assume orientation preservation in the following, i.e., for local injectivity we require that over each triangle t it holds $\det J_f(\xi) > 0$ or equivalently $\det J_\phi(\xi) > 0$, for all $\xi \in T$.

The determinant $\det J_\phi(\xi)$ is a polynomial of degree $\hat{n} = 2(n-1)$. For classical, piecewise linear parametrizations, we thus have $\hat{n} = 0$, i.e. the Jacobian determinant is constant over each triangle – which simplifies the situation significantly: $\det J_\phi$ is easily computed from the signed area of the triangle. Piecewise linear parametrization optimization methods thus commonly perform a line search for every update step, checking that this area stays positive for every triangle [18, 19].

In the case of higher degree, the Jacobian determinant varies over the triangle, and one needs to ensure that it is positive everywhere, i.e. that the *minimum* over the triangle is positive, $\min_{\xi\in T} \det J_\phi(\xi) > 0$. Unfortunately, no closed-form expression is available to compute this minimum. However, one can compute bounds of this minimum determinant from below and from above, allowing to potentially conclude that the map is definitely locally injective or definitely locally non-injective, respectively:

- the lower bound L being positive is a sufficient condition for local injectivity,
- the upper bound U (of the minimum) being positive is a necessary condition for local injectivity.

5.1. Jacobian Determinant Bounds

Obtaining an upper bound of the minimum over T is easy: evaluate the determinant at one or more points (ξ_i), then an upper bound U of the minimum is obviously given by $U = \min_i \det J_\phi(\xi_i)$.

For a lower bound L the situation is more involved. [52, 29] propose to express the determinant in Bernstein-Bézier form. Then, due to this form's convex hull property [63], a lower bound of the value range over the triangle can be read off from the minimum of the Bernstein-Bézier coefficients. This idea was considered by other authors as well, in [53] for arbitrary degree, in [54] for tensor-product maps, or in [36] for tetrahedral meshes.

Recently, [55] described a particularly simple condition based on the orientation of the sub-triangles formed by the Bézier control net. We found it to be faster – but unfortunately, it turns out not to be a sufficient condition, as we show in Appendix D.

To guarantee valid results, we thus make use of the convex hull based technique. [56] elaborate, for the special case $n = 3$, how the tightness of this sufficient condition can be increased arbitrarily by recursive domain subdivision of the Bézier control net. Basis for this is the property of control net convergence to the function graph under subdivision [66]. The same idea is employed by [53]; details on the subdivision strategy are not given.

In contrast to [56], who propose subdivision based on quadrisection, we employ subdivision using repeated bisection [67], cf. Fig. 6. The consequence is a slower rate of refinement, which pays off because most often a very small number of subdivision iterations is necessary. Hence, quadrisection easily leads to some

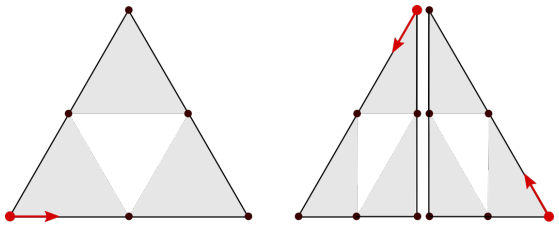


Figure 6: Illustration of bisection of a (quadratic) Bézier triangle. Indexing of the subdivided coefficients is chosen such that the red arrows indicate the first edge of each (sub)triangle, rooted at the $(\hat{n}, 0, 0)$-corner. By this choice of indexing, simply always splitting the first edge leads to proper division of each edge in the course of repeated subdivision.

unnecessary computational cost due to overrefinement. The effect is minor though, as local injectivity of the vast majority of triangles is certified without subdivision in our experiments (see the table in the following section).

We describe the complete algorithm in the following, using the ideas of [52, 29] and spelling out the necessary details for our setting, with bisection based subdivision.

5.2. Injectivity Test Algorithm

The determinant $\det J_\phi(\xi)$ is a polynomial of degree $\hat{n} = 2(n-1)$. In the Bernstein basis, this polynomial reads

$$\det J_\phi(\xi) = \sum_{i+j+k=\hat{n}} d_{ijk} B^{\hat{n}}_{ijk}(\xi),$$

with $\hat{n}(\hat{n}+1)/2$ coefficients $d_{ijk} \in \mathbb{R}$ computed as follows:

$$d_{ijk} = \sum_{\substack{|r|=|s| \\ r+s=(i,j,k)}} \frac{i!\,j!\,k!}{n!\,r!\,s!} (\Delta^0 u_r \Delta^1 v_s - \Delta^1 u_r \Delta^0 v_s),$$

where $r = (r_1, r_2, r_3)$, $|r| = r_1 + r_2 + r_3$, $r! = r_1!r_2!r_3!$, analogously for s, and $\Delta^0 u_r = u_{(r_1+1)r_2r_3} - u_{r_1r_2(r_3+1)}$, $\Delta^1 u_r = u_{r_1(r_2+1)r_3} - u_{r_1r_2(r_3+1)}$, and analogously for v.

Due to the convex hull property $\det J_\phi(\xi) \geq \min_{i+j+k=\hat{n}} d_{ijk}$ for any $\xi \in T$. Due to the corner interpolation property $\det J_\phi(1,0,0) = d_{\hat{n}00}$, and analogously for the other two corners. This leads to the following conditions for local injectivity:

Sufficient condition:. $L = \min_{i+j+k=\hat{n}} d_{ijk} > 0$

Necessary condition:. $U = \min(d_{\hat{n}00}, d_{0\hat{n}0}, d_{00\hat{n}}) > 0$

As there is a gap between these two conditions, there are Bézier triangles for which neither the sufficient condition is satisfied, nor the necessary condition is violated. The local injectivity status can thus not be determined directly using these conditions. To tighten the gap in such cases, we virtually recursively bisect the domain T (cf. Fig. 6), re-expressing the function $\det J_\phi(\xi)$ (in Bernstein-Bézier form) over each of the subdomains using de Casteljau's algorithm [67], until the decision can be made: As soon as the necessary condition is violated in *any* subdomain or the sufficient condition is satisfied in *each* subdomain, the subdivision process can be stopped and the answer returned.

To avoid issues in cases where the minimum determinant is exactly or nearly zero, a maximum recursion depth needs to be set (we use 5). If no decision can be made by then, a violation of local injectivity is reported to stay on the conservative side. As this merely excludes the possibility of getting extremely close to a degenerate state during optimization, this choice is not critical. In particular, only for a very small fraction of all local injectivity test instances during optimization is subdivision necessary at all, as illustrated by these statistics from the optimization of cubic example parametrizations:

	number of tests requiring subdivision depth ...					
	0	1	2	3	4	5+
Foot	1,142,164	3	31	55	15	4
Aircraft	563,482	53	298	93	64	80
Oni	294,618	109	22	29	29	41

The following pseudo code implements this adaptive local injectivity test (in an iterative fashion, with unrolled recursion), to be employed for each triangle in the line search described in Sec. 4.2. The bisection operation is illustrated in Fig. 6 and spelled out in Appendix E.

```
function Is Locally Injective({d_ijk})
    queue Q.push({d_ijk}, 0)
    while Q not empty do
        {d_ijk}, depth ← Q.pop()
        if U({d_ijk}) ≤ 0 return false                ▷ non-injective
        if L({d_ijk}) ≤ 0
            if depth = max_depth return false         ▷ undecided
            {d+_ijk}, {d−_ijk} ← Bézier Bisect({d_ijk})
            Q.push({d+_ijk}, depth+1)
            Q.push({d−_ijk}, depth+1)
    return true                                       ▷ injective
end function
```

6. Global Injectivity

While local injectivity of maps is the property required, e.g., by certain mesh generation or spline construction techniques, for other applications, e.g., texturing, global injectivity is important.

A locally injective map is globally injective (bijective) if the image of the boundary is not self-intersecting [65]. When starting optimization from an initially globally injective map, global injectivity can thus be guaranteed by ensuring that no boundary self-intersections arise, cf. Sec. 2.4.

An efficient approach in the piecewise linear setting is the use of a triangulation S of the ambient space surrounding the image M' of the mesh M in the parameter plane, referred to as scaffold mesh [5, 6]. Optimization is then performed on the combined mesh $M' \cup S$. The idea is that a *global* injectivity violation of M' implies a *local* injectivity violation in $M' \cup S$ in this setting. Hence, the problem of ensuring global injectivity is reduced to that of ensuring local injectivity – already addressed in Sec. 5.

An issue is that the scaffold needs to remain *nice* during optimization, to not significantly restrict the solution space [6]. It is therefore advisable to, after each Newton step, recreate S as a constrained Delaunay triangulation (CDT) [5]. Unfortunately, robust constrained meshing of *curved* domains is an intricate problem [2] (though our recent developments [68] might help).

We avoid this problem by deviating from [5] in two ways when adapting the principle to the higher-order setting:

1. We modify (instead of recreate) the scaffold mesh S, using *Bézier edge flips*, after each optimization step to keep it in good shape. (For the linear, tetrahedral case a modification strategy was employed by [61].) Note that the *initial* scaffold mesh is easily computed as a standard straight-edge CDT, assuming a piecewise linear initialization of M'.
2. Rather than a distortion measure relative to artificial reference elements, we employ a direct triangle shape quality objective $E^S_\triangle$ for the scaffold, better suited to keep it in good shape in the course of optimization.

6.1. Bézier Edge Flip

Following the idea of the classical flip algorithm to turn a triangulation into a Delaunay triangulation [69], we flip an edge of

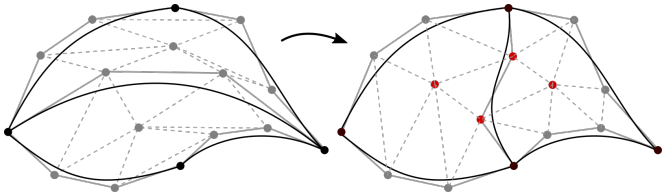


Figure 7: Illustration of edge flip between (here: cubic) Bézier triangles. The interior control points (red) are variable and need to be repositioned such that local injectivity is preserved.

S whenever it violates the local Delaunay property in the linearized version of S. In this linearized version, straight-line triangles are defined by just the triangle corner control points of S. As the (initially straight) scaffold edges empirically do not assume high curvature during optimization, this computationally inexpensive criterion appears to be appropriate for the purpose at hand, though theoretical guarantees are not available.

When flipping an edge in a higher-order mesh, there are degrees of freedom to be settled: while the outer vertex and edge control points of the two involved triangles have to be preserved to maintain a C^0 connection to the surround (cf. Sec. 3.2), the interior control points (red in Fig. 7) can be chosen flexibly – but have to be chosen such that local injectivity is preserved. This is not always possible (with triangles of the same order), but we do not *have to* flip each edge that violates the Delaunay criterion – if valid control point values do not exist or cannot be found, the flip is simply not performed.

We attempt to find valid control points by solving a very small local optimization problem: variables are the interior control point coordinates (red in Fig. 7, i.e., 2 variables for quadratic, 8 variables for cubic cases), starting point is their pre-flip state, and the objective a positive determinant promoting energy [30, Eq. 6].

We note that in previous use cases of edge flips in higher-order simplex settings, the question of map injectivity after the flip was not taken into account. Examples are [62], where the inner control points are set to canonical positions, or [50], where a least-squares fit to the pre-flip function is performed.

6.2. Scaffold Objective

While each triangle of M' has a corresponding reference shape in M relative to which a distortion objective can be formulated, this is not the case for the triangles of S. Using their current state as reference (i.e. promoting preservation of their current shape and size) proved to be a suitable approach for a single optimization step in the piecewise linear case (where the scaffold can be easily recreated after each step) [5]. It, however, proved unsuitable for maintaining a good scaffold quality over tens of iterations.

We therefore rather use a *quality* objective $E_\triangle$ for the scaffold that promotes equilateral elements. We use as $E_\triangle$ a scale invariant conformal distortion objective (referred to as MIPS [70, 71] or Winslow functional [72, §8.2.1]) with respect to an equilateral base element:

$$E^t_{\text{conf}} = (\Sigma^2 + \sigma^2)/\Sigma\sigma = \text{tr}(J_f^\intercal J_f)/\det(J_f). \tag{14}$$

Using the convex-concave decomposition of E_{conf} [4, Eq. 26], this scaffold objective is easily included in the second-order optimization. The following pseudo-code details how the scaffold is combinatorially and geometrically updated after each Newton step. As in [5], the scaffold objective is included with a small factor in the global optimization; we use $\lambda = 10^{-4}\ E^M_{SD}/E^S_\triangle$ (using the objective values from the previous iteration).

```
function OptimizationWithScaffold
    while not converged do
        M' ∪ S ← argmin E^M_SD + λE^S_△ // Newton optimization step
        S ← BézierFlips(S) // Optimize scaffold combinatorially
        S ← argmin E^S_△, M' fixed // Optimize scaffold geometrically
    end while
end function
```

7. Seamlessness

A parametrization is called *seamless* if for each edge e, its images $f_1(e)$ and $f_2(e)$ under the two maps f_1, f_2 associated with the two adjacent triangles are related by a translation plus rotation by some integer multiple of $\pi/2$ [12, 73]. Note that this is a weaker requirement than C^0 continuity (which requires an identity relation), cf. Sec. 3.2. This enables the global parametrization of surfaces with arbitrary topology.

To enable the representation of seamless parametrizations with higher-order per-triangle maps, control points simply must not be shared (cf. Sec. 3.2) across those edges where one requires a non-identity rotation (commonly referred to as *cut edges*).

Let $c_0, \dots, c_n$ be the edge control points along a common cut edge e in adjacent triangle t_1, and $c'_0, \dots, c'_n$ the control points (numbered in the same direction) along this edge in the other adjacent triangle t_2, as illustrated in Fig. 8. As the images $f_1(e)$ and $f_2(e)$ are fully determined by these edge control points, the seamlessness condition can be expressed as

$$(c_i - c_{i+1}) = R^{k\pi/2}(c'_i - c'_{i+1}),\ \forall 0 \le i < n, \tag{15}$$

where $R^{k\pi/2}$ is a (counterclockwise) rotation by angle $k\pi/2$, with $k \in \mathbb{Z}$. Note that the translational component conveniently vanishes when expressing the condition in this differential manner.

7.1. Constraint Incorporation

In the context of piecewise linear parametrization, several previous works have included seamlessness constraints in a soft, penalty based manner [4, 18]. When starting the optimization from a seamless state, however, we can do better by incorporating them as hard constraints, ensuring that they remain satisfied during the (higher-order) optimization.

We assume as input a piecewise linear seamless parametrization, as generated by many recent techniques, such as [11, 12, 73, 74, 75, 76, 77, 78, 79]. To preserve the initial seamlessness during higher-order optimization, we incorporate it as a hard constraint, by eliminating one variable per constraint (15). Note that in this way the PSD property of the Hessian H^M (12) is preserved [80].

In the same elimination manner we can also incorporate hard point constraints to fix selected vertices' initial parameter values.

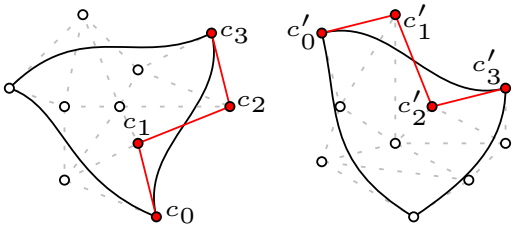


Figure 8: Images of two triangles with a common cut edge under degree 3 maps with their associated control points. The red edge control points here satisfy the seamlessness conditions (15) with $k = 1$.

8. Results

In order to obtain some understanding of the benefits as well as the cost of using higher-order parametrization, we apply the techniques described to a range of models, in various resolutions, in various constraint scenarios, and with various parameter settings (degree, quadrature). Our optimization is generally initialized using an injective piecewise linear parametrization obtained using Tutte's embedding onto a disk of equal area. For experiments with seamlessness constraints (Sec. 8.2), the initial piecewise linear parametrization is computed as described by [12].

In all experiments we employ the local injectivity test from Sec. 5.2 in the line search. Starting from a locally injective initialization, this guarantees that the output is always locally injective, no matter which (potentially local) distortion minimum the optimization ends up in. The effect of additionally ensuring global injectivity is demonstrated in Sec. 8.6.

Note that while the optimization is performed based on $\bar{E}^M$, i.e. quadrature-based approximations of E^M with appropriate numbers of quadrature points (cf. Sec. 8.3), objective values reported and plotted in the following were generally evaluated using a very large number (5050) of quadrature points for accurate and fair comparison. These quasi-ground-truth values are denoted E^M (without the bar).

8.1. Distortion

We ran the described distortion optimization for different degrees n, for a range of models (labeled A, B, ..., Z in the following, specified in Appendix F), in various resolution versions, and using various settings in terms of constraints (free boundary, fixed boundary, pinned vertices). In Fig. 9 we report the final distortion value E^M_{SD}, relative to the base case of degree 1. It can be observed that (not surprisingly), higher order consistently enables achieving lower distortion. The amount of improvement depends on circumstances, though: as can be observed quite clearly, the improvement commonly diminishes with increasing mesh resolution; the benefit is thus particularly significant for coarse meshes. It is important to note that it is not the absolute mesh resolution that determines the gain; rather the mesh resolution relative to the geometric complexity of the surface.

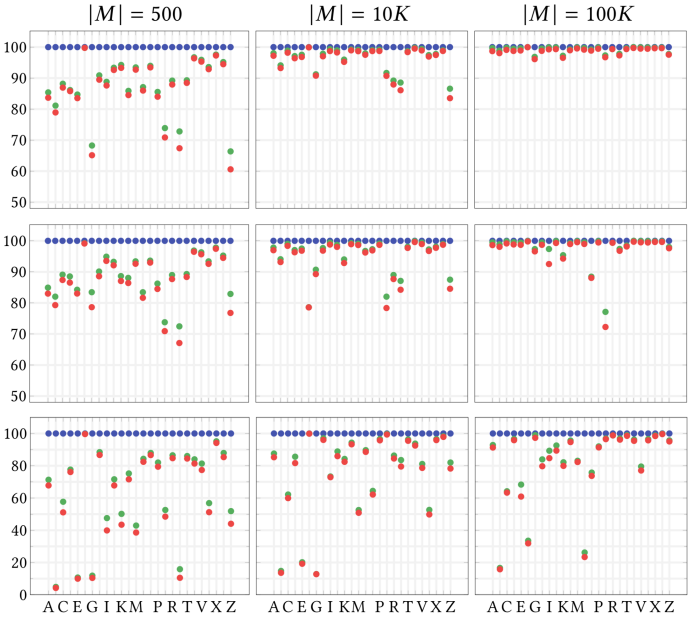


Figure 9: Comparison of final parametrization distortion E^M_{SD} after optimization, for • linear, • quadratic, • cubic order, on 26 different models (A-Z), for three different mesh resolutions (number of triangles, denoted $|M|$), and three different settings: Top row: free boundary. Middle row: fixed boundary. Bottom row: free boundary, but some interior point constraints (five, randomly chosen). The distortion of the linear case is taken as reference (100%), and the other cases are plotted relative to that.

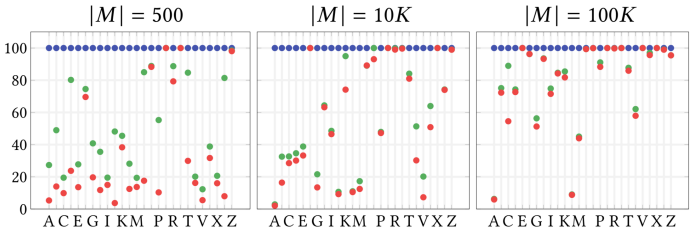


Figure 10: Comparison of final parametrization distortion E^M_{ASAP} after optimization, for • linear, • quadratic, • cubic order, on 26 different models (A-Z), for three different mesh resolutions (number of triangles, denoted $|M|$), with fixed circular boundary. The distortion of the linear case is taken as reference (100%), and the other cases are plotted relative to that.

Conversely, the improvement achieved by higher order parametrization often increases when the parametrization is constrained in a more severe manner (bottom row). Comparing the quadratic and the cubic case, it can also be concluded that going beyond cubic order is likely of very insignificant practical benefit in terms of distortion.

To provide a deeper insight into the distortion differences (beyond the total distortion), we show the histograms of the distortion distribution over the mesh (exemplarily for model C, which shows average behavior in Fig. 9) for these 3×3 settings in Fig. 11.

Analogous to the middle row of Fig. 9, Fig. 10 reports the distortion reduction relative to the linear case when using not an isometry-promoting but a conformality-promoting distortion ob-

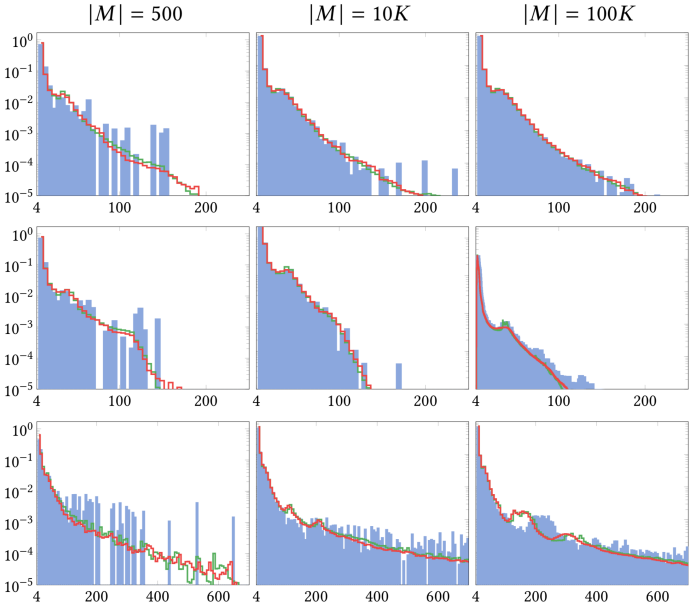


Figure 11: Histograms of final pointwise distortion $E_{SD}(\xi)$, comparing different parametrization orders: • linear, • quadratic, • cubic for model C (for the same settings as in Fig. 9). Pointwise E_{SD} is shown on the horizontal axis, relative surface area on the vertical axis (log scale).

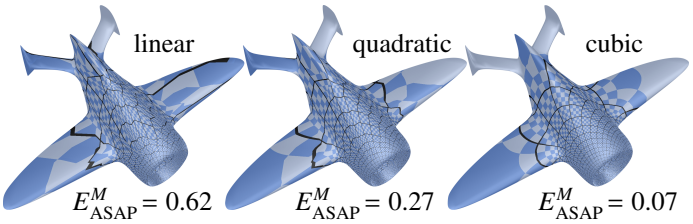


Figure 12: Example result on a coarse mesh using conformal distortion objective $\bar{E}^M_{\text{ASAP}}$. Left: standard piecewise linear parametrization. Middle: piecewise quadratic parametrization. Right: piecewise cubic parametrization. The corresponding final distortion objective values are shown below.

jective $\bar{E}^M_{\text{ASAP}}$, detailed in Appendix B. It can be observed that the benefit is commonly even much larger. This can be attributed to the fact that the isometric distortion objective $\bar{E}^t_{SD}$ favors linearity (rigid maps being the optimum per element) while curved elements can be optimal under $\bar{E}^t_{\text{ASAP}}$ (where more general, non-linear Möbius maps are optimal). Fig. 12 shows an example parametrization.

8.2. Visual Quality

In Fig. 1 an example model (in low (green) and medium (blue) resolution) was textured (see Appendix C for information on the shader-based texture lookup) using a distortion-optimized parametrization with linear or cubic basis functions. An interesting observation is that – while in both cases we ensure only C^0 continuity of the parametrization – there are hardly any texture-isoline kinks noticeable across mesh edges.

In Fig. 13 we added penalty terms to the objective, pulling certain vertices to user-specified positions in parameter space in a 2D deformation example. We show the resulting mesh image in parameter space, comparing the linear and the quadratic case.

In Fig. 14 hard constraints (15) are employed to preserve seamlessness of the linear input parametrization, generated using [12]. Differences are particularly obvious near singularities (parametric cones). Note that relatively coarse meshes have been used for visual clarity here.

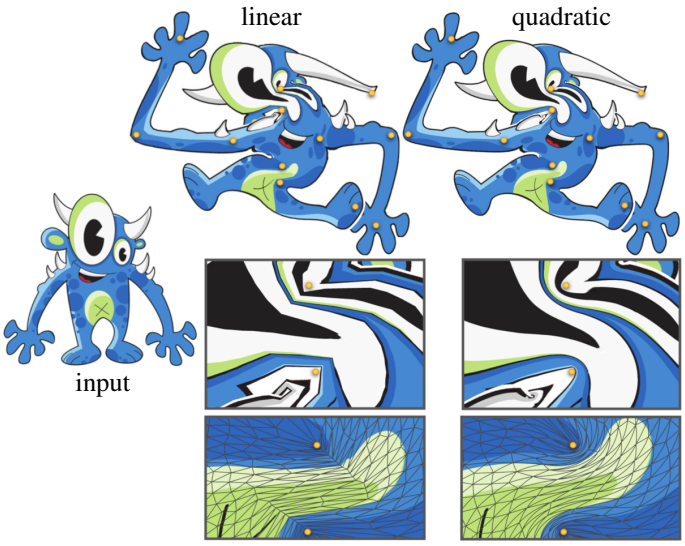


Figure 13: User-specified deformation using soft point constraints (yellow dots), minimizing $\bar{E}_{SD}$. The blow-ups highlight how visible artifacts due to the piecewise linear nature vanish in the quadratic case on the right.

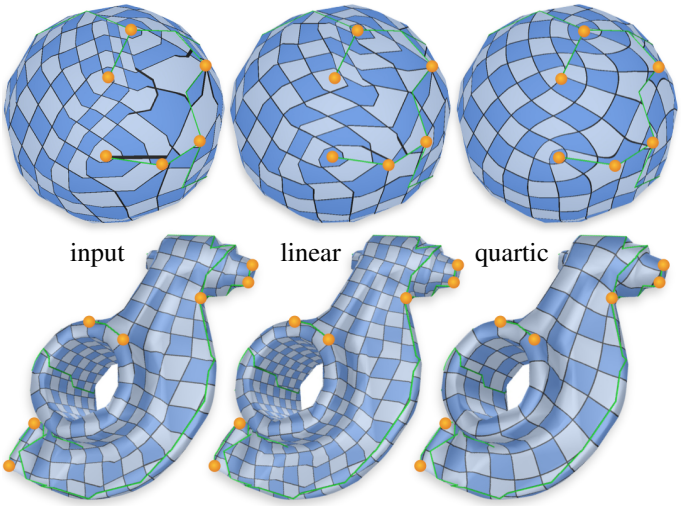


Figure 14: Left: (non-rounded) seamless parametrization generated using the method of [12], taken as starting point by our optimization. Singularities are marked by orange dots. Center: piecewise linear optimization result. Right: piecewise quartic optimization result. Seamlessness is preserved due to constraints (15). Note that seamlessness does not imply the grid matches translationally across cuts (green); to that end the map would furthermore have to be a rounded seamless map [74].

8.3. Choice of Quadrature Scheme

A relevant parameter is the number of quadrature points per triangle in Eq. (9): it determines how accurately the integrated distortion measure is approximated. A choice of too few can slow down convergence and lead into local minima; a generous choice may unnecessarily increase runtime. We tested a range of number choices for various orders. In Fig. 15 the final distortion that is achieved using varying numbers of quadrature points is depicted for the models from Fig. 9 – for the quadratic and cubic case *without* incremental degree elevation. Fig. 16 shows the same *with* incremental degree elevation. It can easily be observed that in this (practically more relevant) case the choice of number of quadrature points is less critical. While relatively small numbers suffice in many cases, we suggest a conservative choice of 36 quadrature points for the quadratic, and 78 for the cubic case.

Note that while with any choice of quadrature scheme the approximation can be arbitrarily off, by construction the parametrization output by our method is always valid (satisfying the desired local/global injectivity properties).

While in previous work on the related topic of curved meshing, e.g. [62, 32, 44], the use of specific quadrature schemes [81, 64] was suggested, we very consistently found equal numbers of uniformly distributed, uniformly weighted quadrature points to perform better in terms of convergence speed and overall optimization behavior (cf. the ◦ marks in Fig. 15). This may be related to the integrand in (9) being a rational function, while the above schemes are optimal for polynomials. We thus use uniform triangular lattice quadrature points, with barycentric coordinates $\frac{1}{m}(a, b, c)$, where $a, b, c \in \mathbb{N}^0$, $a+b+c \leq m$, and uniform weights $w_l = 2/(m(m+1))$. Note that for a choice of $m \in \mathbb{N}$ this leads to $m(m+1)/2$ quadrature points, hence the particular values (3, 6, 10, 15, 21, ...) in Fig. 15 and 16.

8.4. Timing

Our implementation supports arbitrary degree. In a concrete application one could achieve higher performance by specializing to a particular degree. The overhead is not significant, though;

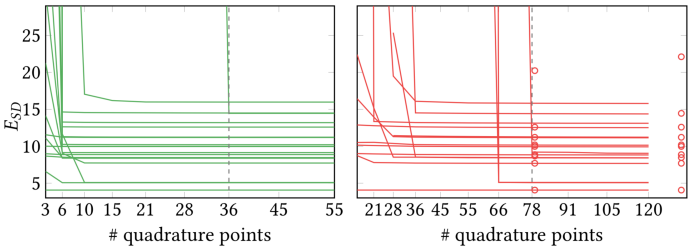


Figure 15: Final distortion E^M_{SD} after optimization using different numbers of quadrature points for • quadratic and • cubic order. Each graph corresponds to one of the models and cases from Fig. 9. The dashed lines indicate our suggested numbers of quadrature points to be used. The ○ marks indicate the final distortion achieved with the quadrature schemes of [64] and [81] for the highest settings reported in these papers (79 and 175 points, respectively) instead of our uniform scheme; some of these marks are not visible due to very high values ($> 10^3$), indicating that optimization got stuck prematurely.

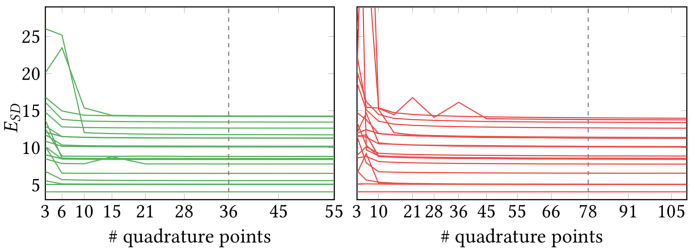


Figure 16: Analogous to Fig. 15 but *with* incremental degree elevation: Final distortion E^M_{SD} after optimization using different numbers of quadrature points for • quadratic and • cubic order.

Fig. 17 shows the time taken for setup and solve by our implementation set to linear, relative to an implementation (*CompMajor*) specifically for the linear case, published by [4].

In Fig. 18 we demonstrate how the numbers of control points and of quadrature points (both increasing with increasing order) affect the time required to perform the optimization (model A, 10K). The behavior is consistent with the expectations that the number of quadrature points has a linear effect on the evaluation cost of energy, Jacobian, and Hessian, while the number of control points (thus the polynomial order) has a super-linear effect.

As outlined in Sec. 4.3, it is thus advisable to start with order 1 optimization and incrementally elevate the degree. In this case essentially any additional time granted (over the amount of time taken by the linear case) immediately leads to advantages in terms of distortion. This is shown in Fig. 19. Reaching the final minimum can, however, still take several times longer.

8.5. Comparison to Refined Piecewise Linear

As has become clear, using higher-order basis functions per element, i.e. per triangle, enables higher quality parametrizations

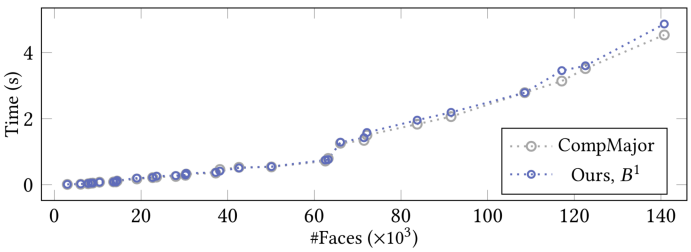


Figure 17: Time (averaged over the optimization iterations) to setup Jacobian and Hessian and to solve for the Newton step (13), relative to the number of faces in different models. We compare our arbitrary-order implementation, set to degree 1, with *CompMajor*, a degree-1-specific optimization code.

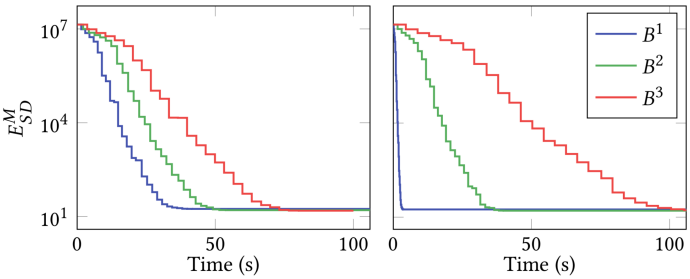


Figure 18: Distortion reduction over time during optimization, comparing • linear, • quadratic, • cubic – all *without* incremental degree elevation; cf. Fig. 19. Left: To highlight the effect of the employed degree (thus the number of control points) on the runtime, here we used a constant (degree-independent) number of quadrature points (55). Right: Using degree-dependent numbers of quadrature points (suggested defaults: B^1: 1, B^2: 36, B^3: 78).

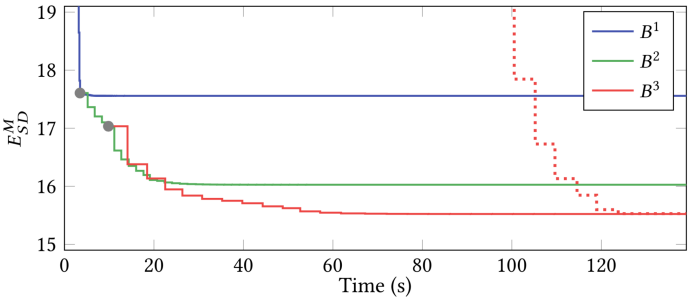


Figure 19: Distortion reduction over time during optimization *with* incremental degree elevation, comparing • linear, • quadratic, • cubic. The dashed graph shows the cubic case *without* incremental degree elevation for comparison.

of lower distortion – at the cost of higher optimization times and the result being in a form that is not standard in geometry processing applications. An intermediate alternative would be to use *piecewise* linear basis functions *per element* – corresponding to classical linear basis functions on a refined version of the mesh – which could simplify interaction with other standard algorithms.

To get an idea of how much refinement would be necessary to achieve comparable result quality and what optimization times would be required on these refined meshes, we performed comparative experiments on hierarchically refined versions of the input meshes. Dyadic refinement, i.e., refinement using 1-to-4 splits, was used, yielding meshes with $4^k|M|$ linear pieces after k refinement iterations applied to an input mesh with $|M|$ triangles.

In Fig. 20 the outcome of this experiment is demonstrated on exemplary input models. It can be observed that with an increasing number of linear pieces per element the final distortion decreases. To achieve a distortion as low as or lower than with *one* quadratic piece per element, sometimes 16 but often at least 64 linear pieces per element (i.e., 2.5 or 7.5 as many degrees of freedom) are necessary. Note that already after one step of refinement (4 pieces per element) the optimization problem has the same size in terms of unknowns as the quadratic case.

We note that other, perhaps adaptive, schemes to determine the number and arrangement of linear pieces per element could be employed; this is a separate research question of potential interest for future work. As the optimally achievable result quality is to some extent determined by the input mesh's quality (in terms of element shape), an interesting particular case would be the use of intrinsic Delaunay overlays [82, 83] to determine the piecewise structure per element. We note, however, that some of the models included in Fig. 9 and 10 that show higher-order benefits (e.g. L, P, U, Z) are nearly intrinsic Delaunay meshes already.

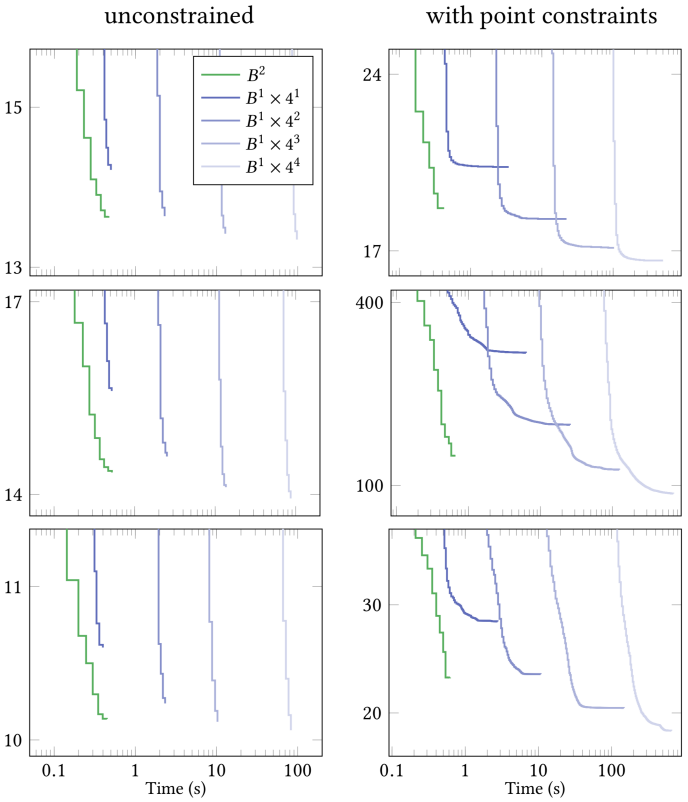
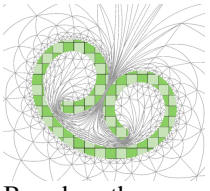


Figure 20: Comparison to refined linear parametrization: energy vs time (doubly logarithmic) during optimization until convergence (obj. decrease < 0.001%) for models A, B, C ($|M| = 500$). Left: free boundary setting of Fig. 9 top. Right: random constrained interior points setting of Fig. 9 bottom.

8.6. Global Injectivity

Examples of using the global injectivity ensuring technique from Sec. 6 are shown, including the higher-order scaffold mesh, in Fig. 21. An interesting piece of statistical information from these experiments is the number of Bézier flips performed. The number of performed flips in these three examples (top to bottom) is 1389, 1127, 460. The number of *intended* flips is 1396, 1141, 469. This means the number of edges that could not be flipped because injectivity preserving inner control point values did not exist or could not be found is very small. The general relevance of Bézier flips for scaffold mesh quality can be observed in the inset, which shows the final result of the Helix example from Fig. 21 if flips were *not* used.



Based on the example from Fig. 13, in Fig. 22 another bijective higher-order example is shown.

8.7. Summary

To briefly summarize the lessons learnt from the experiments:

- Use incremental degree elevation for efficiency. Elevating when the relative change goes < 1% is a good default.
- Use uniform quadrature. Default number of quadrature points per element: B^2: 36; B^3: 78.
- Use quadratic or cubic degree. Going beyond cubic commonly has a negligible effect only.
- The distortion benefit is more significant the more distortion is inevitable (due to constraints or surface curvature).

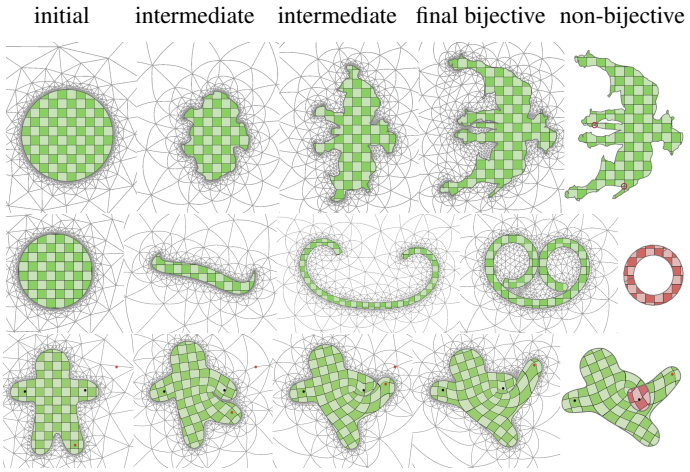


Figure 21: Globally injective parametrization optimization in the quadratic case for three examples (top (Camel) and middle (Helix): free optimization; bottom (Gingerbread): point constraints as in Fig. 13). Shown are the models' images in parameter space, surrounded by the scaffold triangles (white). Left: piecewise linear initialization, followed by two piecewise quadratic intermediate optimization states and the final result; notice the curved scaffold triangles. Right: result when not ensuring global but just local injectivity; overlaps indicated in red. (Camel model from [5])

- The distortion benefit is more significant the lower the mesh resolution (relative to the geometric surface detail).
- The benefit for conformal distortion objectives is larger than for isometric ones.

9. Limitations and Future Work

Efficiency. The main factor responsible for longer computation times compared to linear parametrization is not the increased number of variables (e.g. 6 or 10 control points for degree 2 or 3, compared to 3 for degree 1) but the increased number of summands entering energy, Jacobian, and Hessian due to quadrature (notice the "$M \times Q$" in Eq. (9)). The *setup* of the Hessian matrix, rather than the subsequent *solve* (13), thus easily becomes the bottleneck. Preliminary experiments suggest that significant speed-ups may be achievable by choosing the number of quadrature points adaptively and dynamically, on a per-triangle basis. While generic adaptive quadrature techniques [84] could be employed, investigation of specialized strategies would be interesting. In particular, for triangles over which the Jacobian varies little, fewer quadrature points may be sufficient. This variance could be estimated based on lower and upper bounds of the Jacobian determinant, similar to the approach in Sec. 5.

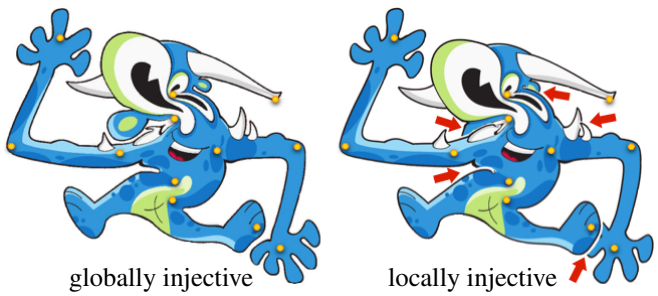


Figure 22: Left: globally injective piecewise quadratic deformation. Right: locally (but not globally) injective piecewise quadratic deformation (from Fig. 13); violations of global injectivity indicated by red arrows.

Continuity. The parametrizations we compute are C^0. Smooth maps of higher continuity may be of interest. Construction of such function spaces (ultimately *spline spaces*) is an involved task on unstructured domains. For instance, to allow achieving C^1 continuity at least quintic polynomials [63, §4.2] or multiple polynomial pieces per triangle [63, §5.1, §5.3] are necessary. Furthermore, initialization and degree elevation are an issue in this context, because the desired level of continuity cannot already be established and maintained throughout the lower degrees in general.

Global Injectivity. The scaffold mesh employed to ensure global bijectivity may, in contrast to the linear case, unnecessarily restrict the solution space when only edge flips are employed for modification. While we have sometimes observed issues with this when using the scaffold element shape preservation objective [5], this was not the case when using our scaffold quality objective $E_{\triangle}$. Deeper investigation of this topic in future work would be valuable.

Inverse Map. In contrast to the linear setting, the inverse f^{-1} of the map f is not readily available in closed form. So while it is easy to map information from the surface into the parameter domain or to pull information from the domain onto the surface, mapping from the domain onto the surface requires non-trivial approximative techniques [85, 86].

Bounded Distortion. Going beyond mere injectivity, strictly bounding distortion may be of interest. Sufficient conditions for the case of spline maps [87, 71] easily adapt to the Bézier case. Unlike in deformation scenarios (as in Fig. 13 and Fig. 22) a major challenge in the case of parametrization lies in finding a valid initialization for optimization.

Finally, our empirical focus herein was on demonstrating that abstract benefits in terms of lower distortion can be achieved using higher-order parametrization. The further exploration of concrete practical advantages that result from this will certainly be of interest.

References

[1] I. Ergatoudis, B. M. Irons, O. C. Zienkiewicz, Curved, isoparametric, "quadrilateral" elements for finite element analysis, Int. Journal of Solids and Structures 4 (1) (1968) 31–42.

[2] Y. Hu, T. Schneider, X. Gao, Q. Zhou, A. Jacobson, D. Zorin, D. Panozzo, Triwild: robust triangulation with curve constraints, ACM Trans. Graph. 38 (4) (2019) 52.

[3] A. W. Bargteil, E. Cohen, Animation of deformable bodies with quadratic Bézier finite elements, ACM Trans. Graph. 33 (3) (2014) 27.

[4] A. Shtengel, R. Poranne, O. Sorkine-Hornung, S. Z. Kovalsky, Y. Lipman, Geometric optimization via composite majorization., ACM Trans. Graph. 36 (4) (2017) 38–1.

[5] Z. Jiang, S. Schaefer, D. Panozzo, Simplicial complex augmentation framework for bijective maps, ACM Trans. Graph. 36 (6).

[6] E. Zhang, K. Mischaikow, G. Turk, Feature-based surface parameterization and texture mapping, ACM Trans. Graph. 24 (1) (2005) 1–27.

[7] M. S. Floater, K. Hormann, Surface parameterization: a tutorial and survey, in: Advances in Multiresolution for Geometric Modelling, Springer, 2005, pp. 157–186.

[8] A. Sheffer, E. Praun, K. Rose, Mesh parameterization methods and their applications, Found. Trends. Comput. Graph. Vis. 2 (2) (2006) 105–171.

[9] L. Kharevych, B. Springborn, P. Schröder, Discrete conformal mappings via circle patterns, ACM Trans. Graph. 25 (2) (2006) 412–438.

[10] Y. Tong, P. Alliez, D. Cohen-Steiner, M. Desbrun, Designing quadrangulations with discrete harmonic forms, Symp. Geom. Proc. (2006) 201–210.

[11] F. Kälberer, M. Nieser, K. Polthier, QuadCover: Surface Parameterization using Branched Coverings, Comp. Graph. Forum 26 (3) (2007) 375–384.

[12] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM Trans. Graph. 28 (3) (2009) 77:1–77:10.

[13] M. Campen, D. Zorin, Similarity maps and field-guided T-splines: a perfect couple, ACM Trans. Graph. 36 (4).

[14] N. Aigerman, R. Poranne, Y. Lipman, Seamless surface mappings, ACM Trans. Graph. 34 (4) (2015) 72:1–72:13.

[15] N. Aigerman, Y. Lipman, Orbifold tutte embeddings, ACM Trans. Graph. 34 (6) (2015) 190:1–190:12.

[16] R. Sawhney, K. Crane, Boundary first flattening, ACM Trans. Graph. 37 (1) (2018) 5.

[17] Y. Lipman, Bijective mappings of meshes with boundary and the degree in mesh processing, SIAM J. on Imaging Sciences 7 (2) (2014) 1263–1283.

[18] M. Rabinovich, R. Poranne, D. Panozzo, O. Sorkine-Hornung, Scalable locally injective mappings, ACM Trans. Graph. 36 (2) (2017) 16.

[19] J. Smith, S. Schaefer, Bijective parameterization with free boundaries, ACM Trans. Graph. 34 (4) (2015) 70:1–70:9.

[20] L. Liu, C. Ye, R. Ni, X.-M. Fu, Progressive parameterizations, ACM Trans. Graph. 37 (4) (2018) 41:1–41:12.

[21] Y. Zhu, R. Bridson, D. M. Kaufman, Blended cured quasi-newton for distortion optimization, ACM Trans. Graph. 37 (4) (2018) 40:1–40:14.

[22] S. Claici, M. Bessmeltsev, S. Schaefer, J. Solomon, Isometry-aware preconditioning for mesh parameterization, Comput. Graph. Forum 36 (5) (2017) 37–47.

[23] S. Z. Kovalsky, M. Galun, Y. Lipman, Accelerated quadratic proxy for geometric optimization, ACM Trans. Graph. 35 (4) (2016) 134:1–134:11.

[24] B. Smith, F. D. Goes, T. Kim, Analytic eigensystems for isotropic distortion energies, ACM Trans. Graph. 38 (1) (2019) 3.

[25] F. Knöppel, K. Crane, U. Pinkall, P. Schröder, Stripe patterns on surfaces, ACM Trans. Graph. 34.

[26] T. DeRose, M. Kass, T. Truong, Subdivision surfaces in character animation, in: Proc. SIGGRAPH '98, ACM, 1998, p. 85–94.

[27] L. He, S. Schaefer, K. Hormann, Parameterizing subdivision surfaces, ACM Trans. Graph. 29 (4).

[28] F. de Goes, M. Desbrun, M. Meyer, T. DeRose, Subdivision exterior calculus for geometry processing, ACM Trans. Graph. 35 (4) (2016) 133.

[29] S. Dey, R. M. O'Bara, M. S. Shephard, Curvilinear mesh generation in 3d, in: 8th Int. Meshing Roundtable, John Wiley & Sons, 1999, pp. 407–417.

[30] J.-F. Remacle, T. Toulorge, J. Lambrechts, J.-C. Weill, Robust untangling of curvilinear meshes, in: Proc. 21st Int. Meshing Roundtable, 2013, pp. 71–83.

[31] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes, Int. Journal for Numerical Methods in Engineering 103 (5) (2015) 342–363.

[32] X. Roca, A. Gargallo-Peiró, J. Sarrate, Defining quality measures for high-order planar triangles and curved mesh generation, in: Proc. 20th Int. Meshing Roundtable, Springer, 2011, pp. 365–383.

[33] A. Gargallo-Peiró, X. Roca, J. Peraire, J. Sarrate, Defining quality measures for mesh optimization on parameterized cad surfaces, in: Proc. 21st Int. Meshing Roundtable, Springer Berlin Heidelberg, 2013, pp. 85–102.

[34] S. J. Sherwin, J. Peiró, Mesh generation in curvilinear domains using high-order elements, International Journal for Numerical Methods in Engineering 53 (1) (2002) 207–223.

[35] E. Ruiz-Gironés, J. Sarrate, X. Roca, Generation of curved high-order meshes with optimal quality and geometric accuracy, Procedia Engineering 163 (2016) 315 – 327, proc. 25th Int. Meshing Roundtable.

[36] P.-L. George, H. Borouchaki, Construction of tetrahedral meshes of degree two, Int. Journal for Numerical Methods in Engineering 90 (9) (2012) 1156–1182.

[37] R. Abgrall, C. Dobrzynski, A. Froehly, A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems, Int. Journal for Numerical Methods in Fluids 76 (4) (2014) 246–266.

[38] Z. Q. Xie, R. Sevilla, O. Hassan, K. Morgan, The generation of arbitrary order curved meshes for 3d finite element analysis, Computational Mechanics 51 (3) (2013) 361–374.

[39] P.-O. Persson, J. Peraire, Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics, 2009.

[40] J. Paul, Orientation preserving mesh optimisation and preconditioning, International Journal for Numerical Methods in Engineering 114 (7) (2018) 749–776.

[41] J. Xu, A. N. Chernikov, Automatic curvilinear quality mesh generation driven by smooth boundary and guaranteed fidelity, Procedia Engineering

82 (2014) 200 – 212.
[42] T. A. Oliver, A high-order, adaptive, discontinuous galerkin finite element method for the reynolds-averaged navier-stokes equations, Ph.D. thesis, Department of Aeronautics and Astronautics (2008).
[43] D. Moxey, D. Ekelschot, U. Keskin, S. J. Sherwin, J. Peiró, High-order curvilinear meshing using a thermo-elastic analogy, Comput. Aided Des. 72 (C) (2016) 130–139.
[44] M. Turner, J. Peiró, D. Moxey, Curvilinear mesh generation using a variational framework, Computer-Aided Design 103 (2018) 73 – 91, 25th International Meshing Roundtable Special Issue: Advances in Mesh Generation.
[45] R. Poya, R. Sevilla, A. J. Gil, A unified approach for a posteriori high-order curved mesh generation using solid mechanics, Computational Mechanics 58 (3) (2016) 457–490.
[46] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X. Li, X. Luo, N. Chevaugeon, J.-F. Remacle, M. W. Beall, R. M. O'Bara, Adaptive mesh generation for curved domains, Applied Numerical Mathematics 52 (2) (2005) 251 – 271.
[47] X.-J. Luo, M. S. Shephard, R. M. O'Bara, R. Nastasia, M. W. Beall, Automatic p-version mesh generation for curved domains, Eng. with Comput. 20 (3) (2004) 273–285.
[48] D. Moxey, M. D. Green, S. J. Sherwin, J. Peiró, On the Generation of Curvilinear Meshes Through Subdivision of Isoparametric Elements, Springer International Publishing, 2015, pp. 203–215.
[49] S. Dey, R. O'Bara, M. Shephard, Towards curvilinear meshing in 3d: The case of quadratic simplices, Computer-Aided Design 33 (2001) 199–209.
[50] D. Cardoze, A. Cunha, G. L. Miller, T. Phillips, N. Walkington, A Bézier-based approach to unstructured moving meshes, in: Proceedings of the 20th Annual Symposium on Computational Geometry, 2004, pp. 310–319.
[51] A. R. Mitchell, G. Phillips, E. Wachspress, Forbidden Shapes in the Finite Element Method, IMA Journal of Appl. Mathematics 8 (2) (1971) 260–269.
[52] X. Luo, M. S. Shephard, J.-F. Remacle, R. M. O'Bara, M. W. Beall, B. A. Szabó, R. Actis, p-version mesh generation issues, in: 11th International Meshing Roundtable, 2002.
[53] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, Journal of Computational Physics 233.
[54] J. Gravesen, A. Evgrafov, D.-M. Nguyen, P. Nørtoft, Planar parametrization in isogeometric analysis, in: Mathematical Methods for Curves and Surfaces, Springer Berlin Heidelberg, 2014, pp. 189–212.
[55] J. Xu, A. Chernikov, A sufficient condition of validity for cubic Bézier triangles, in: Proc. 24th Int. Meshing Roundtable, 2015.
[56] V. Hernandez-Mederos, J. Estrada-Sarlabous, D. L. Madrigal, On local injectivity of 2d triangular cubic Bézier functions, Investigación Operacional 27 (3) (2006) 261–275.
[57] C. Schüller, L. Kavan, D. Panozzo, O. Sorkine-Hornung, Locally injective mappings, in: Proc. Symp. Geom. Proc. 2013, 2013, pp. 125–135.
[58] T. Toulorge, J. Lambrechts, J.-F. Remacle, Optimizing the geometrical accuracy of curvilinear meshes, J. Comput. Phys. 310 (C) (2016) 361–380.
[59] D. Harmon, D. Panozzo, O. Sorkine, D. Zorin, Interference-aware geometric modeling, ACM Trans. Graph. 30 (6) (2011) 137.
[60] P. Jiménez, F. Thomas, C. Torras, 3d collision detection: a survey, Computers & Graphics 25 (2) (2001) 269–285.
[61] M. Müller, N. Chentanez, T.-Y. Kim, M. Macklin, Air meshes for robust collision handling, ACM Trans. Graph. 34 (4) (2015) 133.
[62] L. Feng, P. Alliez, L. Busé, H. Delingette, M. Desbrun, Curved optimal delaunay triangulation, ACM Trans. Graph. 37 (4).
[63] G. Farin, Triangular Bernstein-Bézier patches, Computer Aided Geometric Design 3 (2) (1986) 83–127.
[64] F. D. Witherden, P. E. Vincent, On the identification of symmetric quadrature rules for finite element methods, Computers & Mathematics with Applications 69 (10) (2015) 1232–1241.
[65] O. Weber, D. Zorin, Locally injective parametrization with arbitrary fixed boundaries, ACM Trans. Graph. 33 (4).
[66] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-Spline Techniques, Springer-Verlag, 2002.
[67] J. Peters, Evaluation and approximate evaluation of the multivariate Bernstein-Bézier form on a regularly partitioned simplex, ACM Trans. Math. Softw. 20 (4) (1994) 460–480.
[68] M. Mandad, M. Campen, Bézier Guarding: Precise higher-order meshing of curved 2D domains, ACM Trans. Graph. 39 (4).
[69] M. Bern, D. Eppstein, Mesh generation and optimal triangulation, in: Computing in Euclidean geometry, World Scientific, 1995, pp. 47–123.
[70] K. Hormann, G. Greiner, MIPS: An efficient global parametrization method, Curve and Surface Design '99 (2000) 153–162.
[71] X.-M. Fu, Y. Liu, B. Guo, Computing locally injective mappings by advanced mips, ACM Trans. Graph. 34 (4) (2015) 71:1–71:12.
[72] P. Knupp, S. Steinberg, The Fundamentals of Grid Generation, CRC Press, 1993.
[73] A. Myles, D. Zorin, Global parametrization by incremental flattening, ACM Trans. Graph. 31 (4) (2012) 109.
[74] M. Campen, D. Bommes, L. Kobbelt, Quantized global parametrization, ACM Trans. Graph. 34 (6) (2015) 192.
[75] D. Bommes, M. Campen, H.-C. Ebke, P. Alliez, L. Kobbelt, Integer-grid maps for reliable quad meshing, ACM Trans. Graph. 32 (4) (2013) 98.
[76] A. Myles, N. Pietroni, D. Zorin, Robust field-aligned global parametrization, ACM Trans. Graph. 33 (4) (2014) 135.
[77] X. Fang, H. Bao, Y. Tong, M. Desbrun, J. Huang, Quadrangulation through Morse-parameterization hybridization, ACM Trans. Graph. 37 (4) (2018) 92.
[78] A. Bright, E. Chien, O. Weber, Harmonic global parametrization with rational holonomy, ACM Trans. Graph. 36 (4) (2017) 89.
[79] M. Campen, H. Shen, J. Zhou, D. Zorin, Seamless parametrization with arbitrary cones for arbitrary genus, ACM Trans. Graph. 39 (1).
[80] D. Bommes, H. Zimmer, L. Kobbelt, Practical mixed-integer optimization for geometry processing, in: Int. Conference on Curves and Surfaces, Springer, 2010, pp. 193–206.
[81] S. Wandzurat, H. Xiao, Symmetric quadrature rules on a triangle, Computers & Mathematics with Applications 45 (12) (2003) 1829–1840.
[82] M. Fisher, B. Springborn, P. Schröder, A. I. Bobenko, An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing, Computing 81 (2-3) (2007) 199–213.
[83] N. Sharp, Y. Soliman, K. Crane, Navigating intrinsic triangulations, ACM Trans. Graph. 38 (4) (2019) 55.
[84] J. Berntsen, T. O. Espelid, Algorithm 706: Dcutri: An algorithm for adaptive cubature over a collection of triangles, ACM Trans. Math. Softw. 18 (3) (1992) 329–342.
[85] R. N. Goldman, T. W. Sederberg, D. C. Anderson, Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves, Comput. Aided Geom. Des. 1 (4) (1984) 327–356.
[86] L. Busé, C. D'Andrea, A matrix-based approach to properness and inversion problems for rational surfaces, Appl. Algebra Eng., Commun. Comput. 17 (6) (2006) 393–407.
[87] R. Poranne, Y. Lipman, Provably good planar mappings, ACM Trans. Graph. 33 (4).
[88] B. Lévy, S. Petitjean, N. Ray, J. Maillot, Least squares conformal maps for automatic texture atlas generation, ACM Trans. Graph. 21 (3) (2002) 362–371.
[89] Q. Zhou, A. Jacobson, Thingi10k: A dataset of 10,000 3d-printing models, arXiv preprint arXiv:1605.04797.

Appendix A. Derivative Expressions

Expanding Eq. (11) results in

$$\begin{aligned} H^t = \nabla g^\top \nabla^2 h \nabla g \quad &+ \quad \left(\frac{\partial h}{\partial \Sigma}\right)_+ \nabla^2(\|\alpha\| + \|\beta\|) \\ + \left(\frac{\partial h}{\partial \sigma}\right)_+ \nabla^2\|\alpha\| \quad &- \quad \left(\frac{\partial h}{\partial \sigma}\right)_- \nabla^2\|\beta\|, \end{aligned} \tag{A.1}$$

where

$$\nabla h = 2\begin{bmatrix} \Sigma - \Sigma^{-3} \\ \sigma - \sigma^{-3} \end{bmatrix}, \quad \nabla^2 h = 2\begin{bmatrix} 1 + 3\Sigma^{-4} & 0 \\ 0 & 1 + 3\sigma^{-4} \end{bmatrix}. \tag{A.2}$$

Gradient and Hessian of $\|\alpha\|$ (and entirely analogously $\|\beta\|$) and in turn of g can be computed using

$$\nabla\|\alpha\| = \nabla\alpha^\top \frac{\alpha}{\|\alpha\|}, \quad \nabla^2\|\alpha\| = \frac{1}{\|\alpha\|}\nabla\alpha^\top\left(\mathbf{I} - \frac{\alpha\alpha^\top}{\|\alpha\|^2}\right)\nabla\alpha.$$

Computing $\nabla\alpha$ involves the gradient of $\frac{\partial u}{\partial \xi_0}$, $\frac{\partial u}{\partial \xi_1}$, $\frac{\partial v}{\partial \xi_0}$ and $\frac{\partial v}{\partial \xi_1}$ which can be easily computed as they are linear functions, cf. Eq. (4),

of our variables (the control points). More specifically, using the auxiliary definition $B_{ijk} \equiv 0$ if $i<0$, $j<0$, or $k<0$, the derivatives are given by

$$\frac{\partial}{\partial u_{ijk}}\frac{\partial u}{\partial \xi_0} = n\left(B^{(n-1)}_{(i-1)jk}(\xi) - B^{(n-1)}_{ij(k-1)}(\xi)\right),$$

$$\frac{\partial}{\partial u_{ijk}}\frac{\partial u}{\partial \xi_1} = n\left(B^{(n-1)}_{i(j-1)k}(\xi) - B^{(n-1)}_{ij(k-1)}(\xi)\right).$$

Notice that the Bernstein polynomials need to be evaluated at the quadrature points ξ_l, $l \in Q$, which are the same on all triangles. Thus, for efficiency, we can pre-compute all these relevant Bernstein function values corresponding to our quadrature points once, and reuse them for all triangles and all optimization steps.

Appendix B. Conformal Energy

The conformal distortion objective E_{ASAP} [88] used in Sec. 8.1 in Fig. 10 is defined based on

$$E^t_{\text{ASAP}}(\xi) = (\Sigma(\xi) - \sigma(\xi))^2. \tag{7'}$$

To implement this objective, relative to E_{SD} one merely needs to replace h from (10) by the function

$$h\left(\begin{bmatrix}\Sigma\\ \sigma\end{bmatrix}\right) = (\Sigma - \sigma)^2 \tag{10'}$$

and replace (A.2) with its gradient and Hessian expressions

$$\nabla h = 2\begin{bmatrix}\Sigma - \sigma\\ \sigma - \Sigma\end{bmatrix}, \quad \nabla^2 h = 2\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}. \tag{A.2'}$$

Appendix C. Shader-Based Evaluation

When the higher-order parametrization is to be used for texture mapping, it is convenient to evaluate the mapping functions in a fragment shader. To this end, the per-triangle control points can be made available to the fragment shader via textures or texture buffer objects. Evaluating the texture map for a fragment is then a simple matter of evaluating Eq. (2), involving just powers of the fragment's barycentric coordinates and (precomputed) constant factors $n!/i!j!k!$.

Appendix D. Insufficiency of Control-Triangle Condition

The Bézier control points form a control net consisting of sub-triangles with corners c_{ijk}, $c_{(i+1)j(k-1)}$, $c_{i(j+1)(k-1)}$. The signed area A_{ijk} of such a sub-triangle is

$$A_{ijk} = \frac{1}{2}\det\begin{bmatrix} u_{i(j+1)k} - u_{ijk} & u_{ij(k+1)} - u_{ijk}\\ v_{i(j+1)k} - v_{ijk} & v_{ij(k+1)} - v_{ijk}\end{bmatrix}.$$

The injectivity condition described by [55] is based on the assumption that whenever three pairwise adjacent sub-triangles (depicted grey in Fig. D.23) have positive area, the *de Casteljau* triangle formed by the points with the same barycentric coordinates ξ in each of the three triangles has positive area as well, for any choice of ξ. Fig. D.23 illustrates that this assumption does not hold in general: while in the left configuration, the de Casteljau triangle (green) for $\xi = (1/3, 1/3, 1/3)$ is positively oriented, in the right configuration, it (red) is negatively oriented.

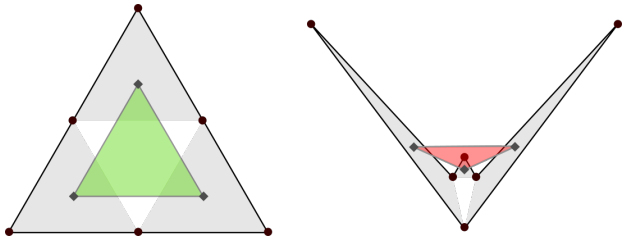


Figure D.23: Counterexample for sufficiency of sub-triangle based condition.

Interpreting the depicted example as a quadratic Bézier triangle, this negative orientation implies that $\det J_\phi(1/3, 1/3, 1/3) < 0$ for the configuration on the right.

While such configurations are rare in unconstrained parametrization settings according to our experiments, in the more intricate constrained setting of Fig. 9 bottom this test actually incorrectly certifies locally non-injective elements at some point during optimization for the majority of the models.

Appendix E. Bézier Bisection

In the following pseudocode we use d_{ij} as a shorthand for $d_{ij(\hat{n}-i-j)}$, where $\hat{n}$ is the degree, and r is a temporary buffer array of length $\hat{n}$.

procedure BÉZIER BISECT($\{d_{ijk}\}$)
  for $0 \le i \le \hat{n}$ **do**
    for $0 \le j \le \hat{n} - i$ **do**
      $r_j \leftarrow d_{ij}$
    $d^+_{0i} \leftarrow r_{\hat{n}-i}$
    $d^-_{0(\hat{n}-i)} \leftarrow r_0$
    for $1 \le l \le \hat{n} - i$ **do**
      for $0 \le j \le \hat{n} - i - 1$ **do**
        $r_j \leftarrow \frac{1}{2}(r_j + r_{j+1})$
      $d^+_{li} \leftarrow r_{\hat{n}-i-l}$
      $d^-_{l(\hat{n}-i-l)} \leftarrow r_0$
  return $\{d^+_{ijk}\}, \{d^-_{ijk}\}$
end procedure

Appendix F. List of Models

Test models A-K (Fig. 9, etc.) were taken from the dataset of [76]. Model L is courtesy of the AIM@Shape repository and the models M-Z were obtained from [89]. Those of disk topology were used directly, in those of sphere topology a hole was cut to obtain a disk topology mesh.

A:	Aircraft	J:	Oni	S:	G_Rebuilt
B:	Armadillo	K:	VaseLion	T:	Hannya
C:	Armchair	L:	Frog	U:	HeadSphinx
D:	Bunny	M:	Aztec	V:	HighresFist
E:	Buste	N:	BigHand	W:	MalteseFalcon
F:	FaceYo	O:	DynamiteBust	X:	Monster
G:	Gargoyle	P:	Dragon	Y:	Plate3
H:	HandOlivier	Q:	Druid	Z:	S_Rebuilt
I:	MaxPlanck	R:	FinialBinary		