Refinable Polycube G-Splines

Martin Sarov[a] and Jörg Peters[a]

[a] *University of Florida*

Abstract

Polycube G-splines form a 2-manifold guided by a mesh of quadrilateral faces such that at most six quads meet at each vertex. In particular, this replicates the layout of the quad faces of a polycube. Polycube G-splines are piecewise bicubic and polycube G-spline surfaces are almost everywhere tangent-continuous (G^1) based on rational linear reparameterization. They can be constructed in two different ways. One construction interprets the quad mesh vertices in the fashion of C^2 bicubic splines – this provides for good shape; the other interprets the 2×2 inner Bézier coefficients of each bicubic as C^1 bicubic B-spline coefficients – this offers four degrees of freedom per patch and enables adaptive refinement so that the resulting G-spline spaces are nested, i.e. any G-spline surface can be exactly re-represented at different levels of refinement.

Keywords: rational linear re-parameterization, nested spaces, geometric continuity, spline surface, polycube, bicubic

1. Introduction

A major challenge when constructing a spline surface from data such as a point cloud or its triangulation, is to determine the patch layout. One approach is to apply an oct-tree-like partition to capture the data as a collection of cubes forming a polycube (see e.g. [THCM04, WYZ+11]). Such polycube approximations address the challenges of axis orientation and associating the data points with (u, v) parameters on the polycube domain.

Starting with a polycube approximation, this paper addresses the second challenge of converting the quadrilateral outer surface of a polycube into a smooth bi-cubic spline surface with the same connectivity. We obtain a surface that

- is almost everywhere G^1 continuous with a *rational linear reparameterization*;
- has, just like regular C^1 bi-cubic splines, *four degrees of freedom (d.o.f.) per patch*, essentially the inner Bézier coefficients;
- affords a refinable, *nested sequence of spaces*, so that the next finer level offers four times as many geometric d.o.f..

The last bullet, deriving a nested sequence of spaces with G-transitions, is the main new contribution of this paper. The construction, abbreviated PGS in the following, improves on a similar construction, [PS15], by relaxing the constraint that all primary vertices, before refinement, be exclusively of valence 3 or 6. Adding vertices of valence 4 and 5 accounts for the vertex configurations occurring in a polycube quad mesh (Fig. 2).

Rational linear reparameterization (when enforcing G^1 continuity) preserves a maximal number of uniformly-distributed additional free parameters at each refinement step. Higher-order reparameterizations imply more constraints that reduce the d.o.f. count, reduce flexibility of the surface where patches join and hence lessen the ability to compute on the surface (see e.g. [CST15]). The d.o.f. of the new bi-cubic construction have a uniform layout: they are akin to the spline coefficients $\mathbb{c}_{ij}$ of bi-cubic B-splines with double knots, i.e. are co-located with the inner four Bézier coefficients $\mathbb{b}_{ij}, 0 < i, j < 3$ of a bi-cubic patch.

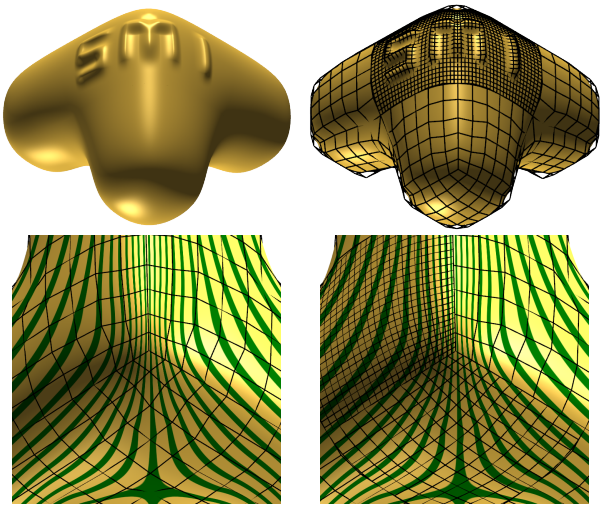

Figure 1: Multi-resolution of Polycube G^1-splines. (Top) After refinement, new local d.o.f. are available for modeling across the G-edges. (Bottom) Exact re-representation of a Polycube Spline surface at different levels of refinement: on the right the Bernstein-Bézier coefficient net is shown at different levels of resolution.

Harvesting all degrees of freedom and creating B-spline-like refinable functions of degree bi-3 associated with the $\mathbb{c}_{ij}$ is non-trivial. Determining the functions requires a careful alternative initialization of the boundary curves via an algorithm for Edge Recovery in Section 4. Unlike [PS15], which built refinement in the spirit of Hierarchical B-splines [FB88] by adding up sur-

faces of several levels, PGS with Edge Recovery provides true nested refinement. That is, a given surface can exactly be re-represented at a finer level as illustrated in Fig. 1 (bottom) and the additional d.o.f. of the adaptive refinement allow modeling features or functions on the surfaces at multiple resolutions Fig. 1 (top).

Overview. After briefly reviewing the literature on polycubes and G^1 constructions with rational linear reparameterizations, Section 2 reviews polycube quad-meshes and basic results on surfaces constructed with rational linear reparameterizations. Section 3 presents PGS as an extension of the construction in [PS15] and discusses its properties. The key section is Section 4 which details the Edge Recovery algorithm that enables local re-generation of boundary curves, yielding nested refinability.

1.1. Literature

Tarini et al. [THCM04] pioneered the use of unions of cubes forming a solid, a polycube, for seamless texture mapping. Mapping each vertex of a closed polygonal surface to a judiciously-chosen polycube, yields a domain for texture coordinates that can reduce distortion compared to mapping from the plane. The tricky construction of good polycubes has been addressed in a series of papers, notably [WHL$^+$08, WJH$^+$08, LJFW08, HWFQ09, XGH$^+$11], with Wang et al. [WHL$^+$08] using the polycube to fit standard splines in the regular mesh regions and filling multi-sided holes with simplex splines. Wan et al. [WYZ$^+$11] optimize polycube domains to minimize the number of cubes while preserving the genus of the original 3D shape. For our purpose a uniform cube partition suffices.

While a number of low-degree, high-quality smooth surface construction exist, we are interested in constructions that have the simplest change of variables between patches. Based on results in [PF10] and [PF09], Peters and Sarov [PS15] argued that, when the tangents at the endpoints are symmetrically distributed, the only way to relate the derivatives of two spline patches abutting along $\mathbf{p}(t,0) = \mathbf{q}(0,t)$ with *linear scaling*, i.e.

$$\begin{aligned} \omega(t)\,\partial_1\mathbf{p}\,(t,0) &= \partial_2\mathbf{p}\,(t,0) + \partial_1\mathbf{q}\,(0,t), \qquad \text{(G1)}\\ \omega(t) &:= (1-t)\omega_0 + t\omega_1, \qquad \omega_0,\omega_1 \in \mathbb{R}, \end{aligned}$$

is that the endpoints $\mathbf{p}(0,0)$ and $\mathbf{p}(1,0)$ have the same valence – or that ω in (G1) is constant, i.e. $\omega_0 = \omega_1$. Restricting the layout such that vertices have valence 3 or 6 or are a partition thereof, Peters and Sarov [PS15] assembled basic configurations of polynomial patches to design a variety of surfaces of genus greater or equal one and smooth with rational linear reparameterization. The present paper generalizes that approach by adding 4-valent and 5-valent vertices with a carefully-chosen non-uniform distribution of emanating curves. Finally, we mention the work of Beccari et al. [BGN14] who introduced rational three-sided macro patches connected with a rational linear reparameterizations.

2. Review of polycube layout and rational linear transition maps

2.1. Polycube vertex classification

We consider a vertex surrounded by a neighborhood $\square$ of $2\times 2\times 2$ cubes aligned with orthogonal directions $\mathbf{d}_i$, $i = 1,2,3$. The cubes are considered either full or empty.

Observation 1 (polycube valences). *The valence n of a polycube vertex is $n := 3 + \min(\eta, \bar\eta)$, where η is the number of directions $\mathbf{d}_j$ for which $\square$ contains cubes on both sides of the plane orthogonal to $\mathbf{d}_j$; and $\bar\eta$ is the number of directions for which $\square$ contains empty cubes on both sides of the plane orthogonal to $\mathbf{d}_j$.*

We will construct smooth approximations of the outer quad-mesh of collections of cubes that are connected by a sequence of shared facets. Non-manifold configurations, e.g. when $\square$ is empty except for two cubes touching at exactly one point, will be treated as separate when constructing surfaces. Analogously, if there are exactly two diagonally opposite empty cubes, two surfaces are fit, one to the 'front' and one to the 'back'. Thus, we can restrict attention to the seven configurations shown in Fig. 2. The only admissible valences for polycube quad surfaces then are 3, 4, 5, or 6.

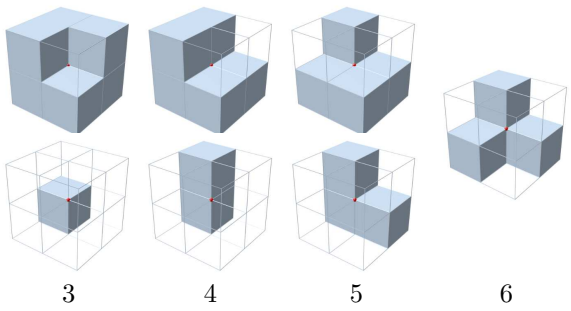

Figure 2: Polycube surface configurations and their valences.

2.2. Indexing the surface quads

Following [PS15] we partition the polycube quads uniformly so that the partition at hierarchy level ℓ yields $2^\ell \times 2^\ell$ sub-quads indexed by the superscript $(k,l) \in 1:2^\ell$ (see Fig. 3) where we used the abbreviation

$$a:b := \begin{cases} a, a+1, \ldots, b, & \text{if } a \le b,\\ a, a-1, \ldots, b, & \text{if } a > b. \end{cases}$$

In general ℓ can be chosen to provide sufficient flexibility for modeling. When focusing on a vertex (corresponding to a cube corner), we associate the superscript $1,1$ with the sub-quads surrounding the point. Each sub-quad (k,l) of an input quad α will be associated with one polynomial patch $\mathbb{b}^{\alpha;kl}$ that we represent in tensor-product Bernstein-Bézier (BB) form of bi-degree $d = 3$ over the unit square $(u,v) \in [0..1]^2$:

$$\mathbb{b}(u,v) := \sum_{i=0}^{d}\sum_{j=0}^{d} \mathbb{b}_{ij} B_i^d(u) B_j^d(v), \qquad (1)$$

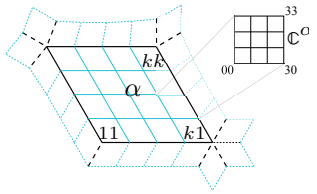

Figure 3: Labeling of quad and the BB-net of control points associated with one sub-quad.

where $\mathbb{b}_{ij}$ are the BB-coefficients and $B_k^d(t) := \binom{d}{k}(1-t)^{d-k}t^k$ is the kth Bernstein-Bézier polynomial of degree d. The BB-coefficients i, j of patch α in sub-quad k, l is then denoted by

$$\mathbb{b}_{ij}^{\alpha;kl}, \quad i, j \in 0:d. \tag{2}$$

2.3. Geometric smoothness constraints

Lemma 1 ([PS15]). *Two patches* $\mathbf{p} := \mathbb{b}^{\alpha,k1}$ *and* $\mathbf{q} := \mathbb{b}^{\beta,1k}$ *of degree bi-3, with BB-coefficients* $\mathbf{p}_{ij}$ *and* $\mathbf{q}_{ij}$*, that share the boundary curve* $\mathbf{p}(t,0) = \mathbf{q}(0,t)$ *join* G^1 *with linear scaling (G1) if and only if*

$$\mathbf{p}_{01} + \mathbf{q}_{10} = \omega_0\mathbf{p}_{10} + (2-\omega_0)\mathbf{p}_{00}, \tag{3}$$
$$\mathbf{p}_{11} + \mathbf{q}_{11} = \boldsymbol{\rho}(\omega_0, \omega_1), \tag{4}$$
$$\mathbf{p}_{21} + \mathbf{q}_{12} = \boldsymbol{\sigma}(\omega_0, \omega_1), \tag{5}$$
$$\mathbf{p}_{31} + \mathbf{q}_{13} = (2+\omega_1)\mathbf{p}_{30} - \omega_1\mathbf{p}_{20}. \tag{6}$$
$$\boldsymbol{\rho}(\omega_0,\omega_1) := \frac{1}{3}\big(2\omega_0\mathbf{p}_{20} - \omega_1\mathbf{p}_{00} + (6-2\omega_0+\omega_1)\mathbf{p}_{10}\big)$$
$$\boldsymbol{\sigma}(\omega_0,\omega_1) := \frac{1}{3}\big(\omega_0\mathbf{p}_{30} - 2\omega_1\mathbf{p}_{10} + (6-\omega_0+2\omega_1)\mathbf{p}_{20}\big).$$

We will add superscripts $\boldsymbol{\rho}^{\alpha;k}$ and $\boldsymbol{\sigma}^{\alpha;k}$ to indicate patch and sub-patch along an edge as we consider the

$$G^1 \textit{ strip of BB-coefficients } \quad \mathbb{b}_{ij}^{\alpha;k1},\ \mathbb{b}_{ij}^{\beta;k1},$$
$$i = 0,1, j = 0,1,2,3, \quad k = 1:2^\ell$$

involved in the G^1 join between two patches $\mathbb{b}^\alpha$ and $\mathbb{b}^\beta$ (see Fig. 4).

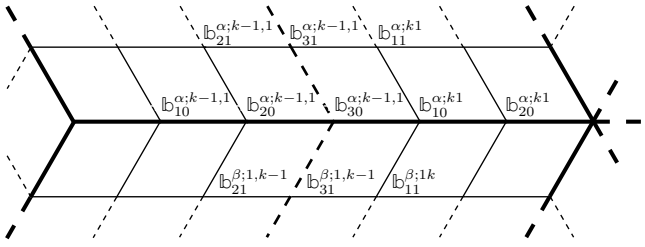

Figure 4: Indices of BB-coefficients of the G^1 strip along a $\langle 3,6\rangle$ edge.

We recall the following lemma from [PS15] restated using the linearity of $\omega(t)$ from (G1).

Lemma 2 ([PS15]). *For* $\mathbb{b}^{\alpha;k-1,1}$ *and* $\mathbb{b}^{\alpha;k,1}$ *to join* C^1*, it is necessary that (see Fig. 4 for the indexing)*

$$\omega(\frac{k}{2^\ell})(-\mathbb{b}_{10}^{\alpha;k-1,1} + 2\mathbb{b}_{20}^{\alpha;k-1,1} - 2\mathbb{b}_{10}^{\alpha;k1} + \mathbb{b}_{20}^{\alpha;k,1}) = 0. \tag{7}$$

i.e. either $\omega(\frac{k}{2^\ell}) = 0$ *or the boundary curve is* C^2*.*

3. G^1 polycube surfaces

We upgrade the algorithm in [PS15] by allowing for vertices of valence 4 and 5.

3.1. Labeling boundary edges by their apparent valence

A simple insight is that a 5-valent vertex can be made to appear to be 6-valent vertex from three directions and to be 4-valent from the remaining two as follows. Let, without loss of generality, $\mathbb{b}_{00}^{\alpha,11}$ be the BB-coefficient shared n surface patches (cf. Fig. 4), and choose local coordinates such that $\mathbb{b}_{00}^{\alpha,11}$ is the origin. We say that the vertex of valence n *appears to be of valence* p *for an edge* e between patches α and $\alpha+1$, if the tangent coefficients $\mathbb{b}_{10}^{\beta,11}$, $\beta \in \{\alpha-1, \alpha, \alpha+1\}$ (modulo n) satisfy

$$2c_p\ \mathbb{b}_{10}^{\alpha,11} = \mathbb{b}_{10}^{\alpha+1,11} + \mathbb{b}_{10}^{\alpha-1,11}, \quad c_p := \cos\frac{2\pi}{p}. \tag{8}$$

Specifically, when $n = 5$, we temporarily insert a dummy edge (dashed edge in Fig. 5a) to obtain tangent coefficients $\mathbb{b}_{10}^{\alpha,11}$, $\alpha \in 0:5$ and project these onto an affine image of a circle with center $\mathbb{b}_{00}^{\alpha,11}$. Removing the dummy tangent, we are left with $n = 5$ tangent coefficients $\mathbb{b}_{10}^{\alpha,11}$, $\alpha \in 0:4$ and opening angles of $[1,1,1,1,2]\ 60°$. Three of these edges, e_1, e_2, e_3, satisfy (9), hence receive the label 6, while the other two, e_0, e_4 satisfy (10), and receive the label 4:

$$2c_6\ \mathbb{b}_{10}^{\alpha,11} = \mathbb{b}_{10}^{\alpha,11} = \mathbb{b}_{10}^{\alpha+1,11} + \mathbb{b}_{10}^{\alpha-1,11}, \tag{9}$$
$$2c_4\ \mathbb{b}_{10}^{\alpha,11} = 0 = \mathbb{b}_{10}^{\alpha+1,11} + \mathbb{b}_{10}^{\alpha-1,11}, \quad c_n := \cos\frac{2\pi}{n}. \tag{10}$$

In a polycube configuration (cf. the third column of Fig. 2), the second of the three edges with label 6, e_2, is by *default assigned* to be the edge towards the single neighbor that is not the plane shared by the other four. Analogously, the edges of a 4-valent vertex can be assigned labels $3,4,6,4$ as shown in Fig. 5b so that the vertex is a part of a longer edge between a vertex of valence 6 and a vertex of valence 3 (and thereby avoids edges connecting valence 4 and valence $m \neq 4$).

For edge sequences with labels $\langle m,4\rangle\ \langle 4,n\rangle$ where $m \neq n$, we will only guarantee C^0, not G^1 or C^1 continuity. Appendix A specifies a pre-processing Edge Labeling algorithm that pre-labels the input quad mesh to minimize the number of such edge pairs.

3.2. PGS: Constructing a smooth, piecewise bi-3 surface from a polycube

The PGS algorithm subsumes the algorithm in [PS15].

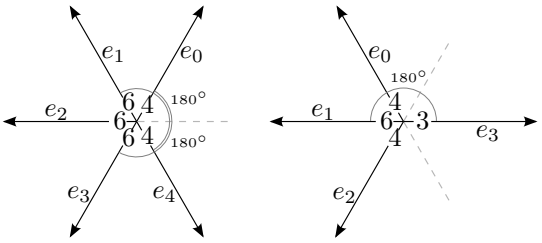

Figure 5: Apparent configuration of tangents emanating from vertices of valence $n = 5, 4$ based on symmetrically distributing $n = 6$ tangents and removing tangents.

For an $\langle n, m\rangle$ edge, we abbreviate

$$\omega_{ij}^k := \omega(\frac{k}{2^\ell}), \quad \omega(t) := 2(1-t)c_n - tc_m,\ c_n := \cos(\frac{2\pi}{n}),$$

and define $\mathbb{1}$ be a row vector of ones, $\mathbb{b}_{ij}$ the vector with entries $\mathbb{b}_{ij}^{\alpha;11}$ according to counterclockwise (ccw) ordering, and $\mathbb{b}_{ij}^+$ the vector same vector except with entries shifted ccw by one.

We initialize as follows. Edge Labeling labels all edges so that the labels are 3,4, or 6. One Catmull-Clark subdivision step is applied to the input mesh. Interpreting the resulting quad mesh points as bi-3 B-spline control points generates default degrees of freedom (d.o.f.) that form a collection of B-spline-like control points $\mathbb{c}$ for the $G-$splines. Specifically, the rules for conversion from B-spline to BB-form define a (collection of non-boundary) BB-coefficients

$$\mathbb{b}^\circ := \{\mathbb{b}_{ij}^{\alpha;kl}, \quad i, j \in \{1, 2\}\}.$$

(The conversion rule is $\mathbb{b}_{11}^{\alpha;kl} = 4\mathbf{v}_{00}+2\mathbf{v}_{10}+2\mathbf{v}_{01}+\mathbf{v}_{11})/9$ for four B-spline control points $\mathbf{v}_{ij}$ forming the quad.) For a regular grid layout the points in $\mathbb{b}^\circ$ coincide with (but have different basis functions than) the coefficients of C^1 bicubic splines with double knots.

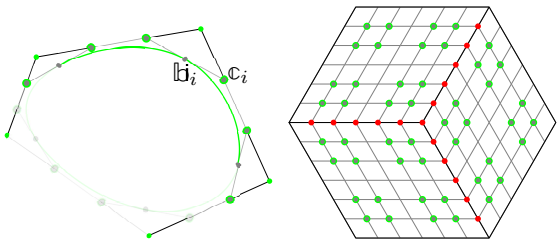

Figure 6: B-spline-like control points $\mathbb{c}$ (green) and dependent BB-coefficients obtained by C^1 constraints (gray) or G^1 constraints (G1) in red. (a) analogous curve case; (b) G^1 spline d.o.f.: due to the initialization by one Catmull-Clark step, there are four patches per polycube quad and each has 2×2 control points $\mathbb{c}$.

We initialize $\mathbb{c} := \mathbb{b}^\circ$. There are exactly 2×2 such control points per quad of the Catmull-Clark refined polycube mesh (see Fig. 6). We can now state the PGS algorithm.

PGS algorithm: (Polycube G-spline algorithm)

Input: $4N$ B-spline-like control points $\mathbb{c}$, 4 for each of the N facets of a quad mesh;
a refinement level ℓ.

Output: A surface consisting of bi-cubic tensor-product spline surfaces $\mathbb{b}^\alpha$ that join G^1 – except along edges labeled $\langle 4, n\rangle$ where the surface is only guaranteed to be C^0.

Overview: Visiting each sub-patch $\mathbb{b}^{\alpha;k1}$, $\mathbb{b}^{\beta;k1}$ corresponding to the unsubdivided (polycube) quad edge, we start from each $\langle 6, m\rangle$ edge and work along the original edge to the middle index, $k = 1 : \mu_\ell$ where $\mu_\ell := 2^{\ell-1}$ is half-ways to its neighbor. In the process we adjust the BB-coefficients to form a G^1 strip. Then we complete the input half edges starting from outgoing labels 4 and finally we complete the remaining original edges from the middle towards the vertices of valence 3.

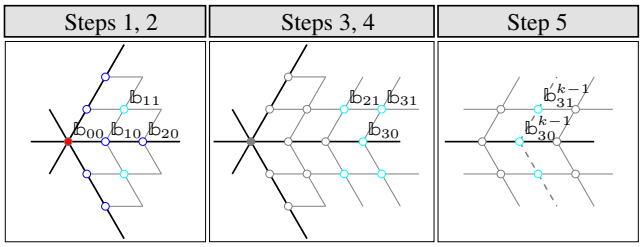

Figure 7: Steps of PGS (for each label in {6,4,3}).

Algorithm:

0. Set all inner coefficients $\mathbb{b}_{kl}^{\alpha;ij} := \mathbb{c}_{kl}^{\alpha;ij}$ for all subpatches with indices $\alpha; ij$ and $0 < k, l < 3$. (In the following, we leave out parts or all of the superscript $\alpha; ij$ whenever clear by context. For example, we abbreviate the above as $\mathbb{b}_{kl} := \mathbb{c}_{kl}$.)
   Set all patch corner points to the average of the surrounding n interior coefficients

$$\mathbb{b}_{00} := \frac{1}{n}\mathbb{1}\mathbb{b}_{11} = \sum_{\alpha=0}^{n-1} \mathbb{b}_{11}^\alpha. \tag{11}$$

   Set the common boundary coefficients of neighbor subpatches α and β as the symmetric average of their neighboring inner coefficients:

$$\mathbb{b}_{l0}^\alpha := \frac{1}{2}(\mathbb{b}_{l1}^\alpha + \mathbb{b}_{l1}^\beta). \tag{12}$$

   For $n = 3$, adjust for better shape

$$\mathbb{b}_{10} \leftarrow \mathbb{b}_{10} + \frac{1}{2}(\mathbb{b}_{10} - \mathbb{b}_{00}).$$

For all edges labeled $\langle 6, m\rangle$ (then for $\langle 4, m\rangle$ and finally for $\langle 3, m\rangle$),

1. For $n = 5$ temporarily insert a dummy tangent between the edges with first label 4 (e_0 and e_4 in Fig. 5). For $n \in \{5, 6\}$, for all patches α surrounding the center vertex, apply the projection matrix P to the tangent coefficients

$$\mathbb{b}_{10} \leftarrow P\,\mathbb{b}_{10}, \qquad P := (\mathbf{1} + C)/n \in \mathbb{R}^{n\times n} \tag{13}$$
$$\mathbf{1}(i, j) := 1, \quad C(i, j) := 2c_{j-i}.$$

   (For $n = 5$, P ensures that edges e_0,and e_3 respectively e_1 and e_4 are collinear.) For $n = 5$ ignore the dummy entry

in $\mathbb{b}_{10}$.
For $n = 4$, set (cf. (4) with $\omega_0 = 0$)

$$\mathbb{b}_{10} := \frac{3(\mathbb{b}_{11} + \mathbb{b}_{11}^{+}) + \omega_1 \mathbb{b}_{00}}{6 + \omega_1}. \tag{14}$$

2. Let $\mathbf{r}_\alpha := \boldsymbol{\rho}^{\alpha;1}(\omega_0^1, \omega_1^1)$ and $\mathbf{r} \in \mathbb{R}^{n\times 3}$ the vector with entries $\mathbf{r}_\alpha$ in counterclockwise order. If $n = 6$, adjust the second derivatives at the vertex

$$\mathbb{b}_{20} \leftarrow \mathbb{b}_{20} - \frac{(-1)^\alpha}{4}\mathbf{a}, \quad \mathbf{a} := \sum_{\alpha=0}^{n-1}(-1)^\alpha \mathbf{r}_\alpha, \tag{15}$$

If $n \neq 4$, form the $n \times n$ circulant matrix M

$$M(i,j) := \begin{cases} 1 & \text{if } i = j \text{ or } i = j-1 \mod n \\ 0 & \text{else} \end{cases}.$$

and denote by M^- the pseudo-inverse of M when $n = 6$ and M^{-1} otherwise. Set

$$\mathbb{b}_{11} := M^- \mathbf{r}. \tag{16}$$

3. For each $\langle 6, m\rangle$ edge, where $\omega \neq 0$, make the curves C^2 up to the *middle of the edge*. For $k = 1 : \mu_\ell$

$$\mathbb{b}_{10}^{\alpha;k+1,1} := \mathbb{b}_{20}^{\alpha;k1} + \frac{1}{2}(\mathbb{b}_{20}^{\alpha;k+1,1} - \mathbb{b}_{10}^{\alpha;k1}). \tag{17}$$

For each $\langle 3, m\rangle$ edge starting from the midpoint, make the curves C^2. For $k = \mu_\ell : 1$ (note the ordering!)

$$\mathbb{b}_{20}^{\alpha;k1} := \mathbb{b}_{10}^{\alpha;k+1,1} - \frac{1}{2}(\mathbb{b}_{20}^{\alpha;k+1,1} - \mathbb{b}_{10}^{\alpha;k1}). \tag{18}$$

Observe that $\langle 3, 6\rangle$ edges are executed only after $\langle 6, 3\rangle$ edges.

4. For $k = 1 : \mu_\ell$, i.e. up to the middle of the edge, minimally perturb the transversal derivatives to enforce G^1 continuity:

$$\mathbb{b}_{11}^{\alpha;k1} \leftarrow \mathbb{b}_{11}^{\alpha;k1} + \frac{1}{2}\big(\boldsymbol{\rho}^{\alpha;k}(\omega_0^k, \omega_1^k) - \mathbb{b}_{11}^{\alpha;k1} - \mathbb{b}_{11}^{\beta;1k}\big), \tag{19}$$

$$\mathbb{b}_{21}^{\alpha;k1} \leftarrow \mathbb{b}_{21}^{\alpha;k1} + \frac{1}{2}\big(\boldsymbol{\sigma}^{\alpha;k}(\omega_0^k, \omega_1^k) - \mathbb{b}_{11}^{\alpha;k1} - \mathbb{b}_{11}^{\beta;1k}\big),$$

5. For $k = 1 : \mu_\ell$ average up to the midpoint (to enforce C^1 continuity within the 3-6 quads α and the neighbor β of α):

$$\begin{aligned} \mathbb{b}_{30}^{\alpha;k-1,1} &:= (\mathbb{b}_{20}^{\alpha;k-1,1} + \mathbb{b}_{10}^{\alpha;k1})/2, \\ \mathbb{b}_{31}^{\alpha;k-1,1} &:= (\mathbb{b}_{21}^{\alpha;k-1,1} + \mathbb{b}_{11}^{\alpha;k1})/2, \\ \mathbb{b}_{31}^{\beta;1,k-1} &:= (\mathbb{b}_{21}^{\beta;1,k-1} + \mathbb{b}_{11}^{\beta;1k})/2. \end{aligned} \tag{20}$$

This completes the PGS algorithm.

3.3. Examples of PGS surfaces

Examples of surfaces obtained from various polycube configurations are shown in Fig. 8 and Fig. 12. In particular, the test cases in Fig. 8 include vertices of valence 5 surrounded by vertices of valence 3,4,5,6 labeled so that the surfaces are smooth along these edges.

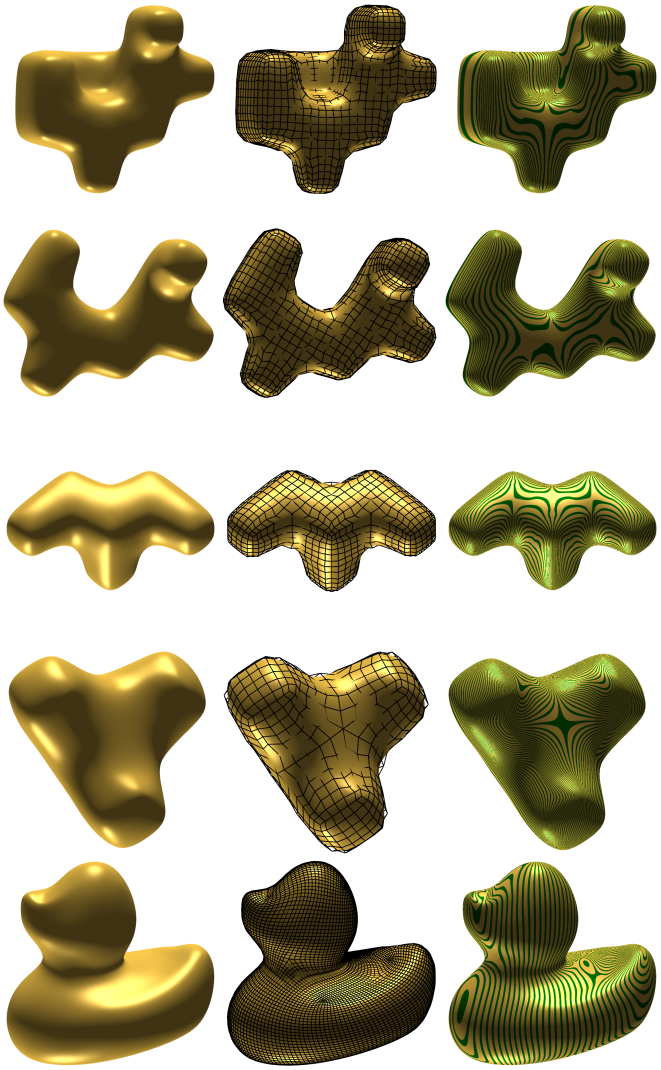

Figure 8: Smooth PGS surfaces (without refinement) from polycube configurations. Form left to right: shaded image (two light sources), BB-control net, highlight lines.

3.4. Properties of the surfaces

Since ω is constant for $\langle 3, 6\rangle$ edges, the corresponding pairs of tensor-product spline patches joining across a $\langle 3, 6\rangle$ edge are parametrically C^1 connected. Across equally labeled edges ($\langle 3, 3\rangle$, $\langle 4, 4\rangle$, $\langle 6, 6\rangle$) the patches are G^1 connected.

According to [PF09], we cannot hope for smoothness when ω is linear and we have opposing $\langle n, 4\rangle$ and $\langle 4, m\rangle$ edges forming a label sequence $\langle n, 4\rangle$, $\langle 4, m\rangle$ with $n \neq m$. The subtle challenge is that if $n \neq m$ then the lengths of the tangents of the transversal curve with coefficients $\mathbb{b}_{3i}^{\alpha;11}$, $i = 0 : 3$,

$$\begin{aligned} &\mathbb{b}_{30}^{\alpha;11} - \mathbb{b}_{20}^{\alpha;11} = \mathbb{b}_{10}^{\alpha+1;11} - \mathbb{b}_{30}^{\alpha+1;11}, \text{ but} \\ &\mathbb{b}_{31}^{\alpha;11} - \mathbb{b}_{21}^{\alpha;11} = \nu(\mathbb{b}_{11}^{\alpha+1;11} - \mathbb{b}_{31}^{\alpha+1;11}), \text{ where } \nu \neq 1. \end{aligned}$$

do not agree when ω is linear. Nevertheless sequences $\langle n, 4\rangle$, $\langle 4, m\rangle$ with $n \neq m$, often do not visibly disrupt smoothness: highlight line discontinuities are very small when detected, see Fig. 9. Also, when

the partial derivatives of the transversal curve are all collinear, then the surface is smooth since the determinant $\det[(\partial_1 \mathbb{b}^{\alpha;k1})(t,0), (\partial_2 \mathbb{b}^{\alpha;k1})(t,0), (\partial_1 \mathbb{b}^{\alpha;k+1,1})(t,0)]$ along the transversal boundary vanishes.

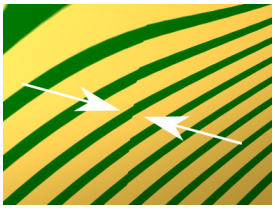

Figure 9: Highlight line mismatch visible under zoom (1/100 th of the edge length).

In preparation for refinement we note the following.

Lemma 3. *The placement of the vertex coefficients* $\mathbb{b}_{00}^{\alpha;11}$ *by PGS is identical to [PS15] where* $\mathbb{b}_{00}^{11} := \frac{1}{n}\mathbb{1}P\,\mathbb{b}_{10}^{11}$.

Proof Denote by $\mathbb{b}_{ij}$ the vector of $\mathbb{b}_{ij}^{\alpha;11}$ surrounding the valence n vertex and by $\mathbb{b}_{ij}^{+}$ the vector $\mathbb{b}_{ij}$ with the indices incremented by one. The initialization by B-spline to BB-form conversion defines $\mathbb{b}_{10} := (\mathbb{b}_{11} + \mathbb{b}_{11}^{+})/2$. Since $\mathbb{1}C = [0,\ldots,0]$, and $\mathbb{1}\mathbf{1} = n\mathbb{1}$, the claim follows from

$$\begin{aligned}\mathbb{1}P\,\mathbb{b}_{10} &= \mathbb{1}\mathbf{1}\mathbb{b}_{10}/n + \mathbb{1}C\mathbb{b}_{10}/n = \mathbb{1}\mathbb{b}_{10} \\ &= \mathbb{1}(\mathbb{b}_{11} + \mathbb{b}_{11}^{+1})/2 = \mathbb{1}\mathbb{b}_{11}.\end{aligned} \tag{21}$$

|||

4. Refinement

A given surface is unchanged when splitting the patches via deCasteljau's algorithm. But the resulting finer-level Bézier coefficients are clearly not all d.o.f. of the G^1 polycube spline space. Equally obvious, partitioning the polycube quads to serve as input would lead to different surfaces – for example the patches corresponding to interior polycube facets would be flat. One can add surfaces at different levels of refinement as in [PS15] (akin to Hierarchical B-splines [FB88]), but for applications it is more convenient to have an adaptive but uniform representation of the d.o.f. without having to add up functions of all levels.

We would like a subdivision algorithm for the d.o.f. where the next level is obtained as a local convex combination of the d.o.f. of the previous level. Can one, say by binary partition as in the Catmull-Clark algorithm, derive a new B-spline-like control net so that application of PGS yields the same surface? We think this is not possible, already due to the initial 2×2 split across which the pieces only join C^1 – whereas the model for the initialization of PGS is conversion of a C^2 bicubic B-spline surface to Bézier form.

However a refinement can be obtained in analogy with bicubic B-splines with *double knots* and observing that, for C^1 bi-3 splines, the B-spline control points and the inner control points

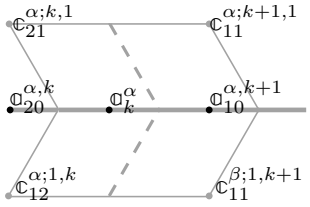

Figure 10: Averages of polycube spline coefficients used in the Edge Recovery algorithm.

$\mathbb{b}^{\circ} := \{\mathbb{b}_{ij}^{\alpha;kl}, 0 < i,j < 3\}$ of the BB-form coincide. So we could set $\mathbb{c} := \mathbb{b}^{\circ}$ and apply B-spline subdivision. This seems almost too easy a solution to the refinement problem, and it is: PGS does not depend exclusively on the $\mathbb{b}^{\circ}$ but, in Step 3, also uses information from the boundary curves. The deal breaker is that Step 4 then perturbs the $\mathbb{b}^{\circ}$ based on the newly-computed boundary curves.

So we return to De Casteljau's algorithm, but focus on mapping control points $\mathbb{b}^{\circ,\ell}$ at level ℓ to $\mathbb{b}^{\circ,\ell+1}$ at level $\ell+1$:

$$\mathbb{b}_{ij}^{\circ,\ell+1} := \sum_{kl} a_{kl}^{\ell}\mathbb{b}_{kl}^{\circ,\ell}. \tag{22}$$

Unlike B-spline refinement stencils, the a_{kl}^{ℓ} involve, via de Casteljau's algorithm, a number of BB-coefficients defined by the G^1 constraints.

4.1. Invariant reconstruction of boundary curves

To apply de Casteljau's algorithm to the level ℓ d.o.f. $\mathbb{c}^{\ell}$, we need to derive boundary curves from the $\mathbb{c}^{\ell}$ so that Steps 1-5 of PGS will reproduce the same surface. Then we can apply de Casteljau's algorithm to obtain four times as many d.o.f. $\mathbb{c}^{\ell+1}$ and regenerate the exact same surface by Steps 1-5 of PGS. Of course when the d.o.f. are moved, Steps 1-5 yield a new and different smooth surface at the refined level.

The construction of boundary curves in the interior and for all $\langle 4,4\rangle$ edges is trivial, namely averaging $\mathbb{c} = \mathbb{b}^{\circ}$ to enforce C^1 transitions. The challenge is that the G^1 constraints (3)–(6) link boundary coefficients all along the boundaries, resulting in a global system.

The key to a local construction is (7) of Lemma 2. When $\omega(k/2^{\ell}) = 0$ then we can locally apply C^1 constraints to see that

$$\mathbb{b}_{30}^{\alpha;k,1} = (\mathbb{b}_{21}^{\alpha,k,1} + \mathbb{b}_{21}^{\beta,1,k} + \mathbb{b}_{11}^{\alpha,k,1} + \mathbb{b}_{11}^{\beta,1,k})/4 \tag{23}$$

and we can unravel the linked constraints from the middle $k = \mu_{\ell} - 1$ of the subdivided boundary curve back to the vertex with outgoing label $m \neq 4$. Similarly, when $\omega(k/2^{\ell}) \neq 0$, the boundary curves have to be C^2. These C^2 constraints on the curve together with Lemma 3 yield a 3×3 system of equations at the endpoints. This system is easily solved (see (24) below) and forms the start to locally solve for the boundary coefficients. The result is the following algorithm for Edge Recovery.

Edge RecoveryAlgorithm
We abbreviate (cf. Fig. 10)

$$\begin{aligned}\mathbb{o}_{20}^{\alpha,k} &:= (\mathbb{c}_{21}^{\alpha;k,1} + \mathbb{c}_{12}^{\alpha;1,k})/2,\\ \mathbb{o}_{10}^{\alpha,k+1} &:= (\mathbb{c}_{11}^{\alpha;k+1,1} + \mathbb{c}_{11}^{\beta;1,k+1})/2,\\ \mathbb{o}_{k}^{\alpha} &:= (\mathbb{o}_{10}^{\alpha,k+1} + \mathbb{o}_{20}^{\alpha,k})/2.\end{aligned}$$

Input Spline coefficients $\mathbb{c}_{ij}^{\alpha,k,\ell}$, $i,j \in \{1,2\}$ of a partitioned polycube quad mesh, four per sub-quad.
Output Cubic boundary curves suitable for PGS.
Algorithm (i) *Initialize* the vertex BB-coefficients

$$\mathbb{b}_{00}^{\alpha;11} := \frac{1}{n}\sum_{\alpha=0}^{n-1} \mathbb{b}_{11}^{\alpha;11}. \tag{10'}$$

(ii) Where $\omega_1^{k^*} = 0$ (note that $|\omega_0^k| \le 1$), for $k = k^*$

$$\mathbb{b}_{30}^{\alpha;k,1} := \mathbb{o}_k^{\alpha}, \tag{6'}$$

$$\mathbb{b}_{20}^{\alpha;k,1} := \frac{1}{6-\omega_0^k}(6\mathbb{o}_{20}^{\alpha,k} - \omega_0^k \mathbb{b}_{30}^{\alpha;k,1}), \tag{5'}$$

$$\mathbb{b}_{10}^{\alpha;k,1} := \frac{1}{6-2\omega_0^k}(6\mathbb{o}_{10}^{\alpha,k} - 2\omega_0^k \mathbb{b}_{20}^{\alpha;k,1}), \tag{4'}$$

and for $k := k^*-1,\ldots,1$ (when $\omega_1^k \neq 0$)

$$\mathbb{b}_{30}^{\alpha;k,1} := \frac{1}{2-\omega_1^k}(\mathbb{o}_k^{\alpha} - \omega_1^k \mathbb{b}_{10}^{\alpha;k+1,1}), \tag{3k+'}$$

$$\mathbb{b}_{20}^{\alpha;k,1} := \frac{1}{-\omega_1^k}(\mathbb{o}_k^{\alpha} - (2+\omega_1^k)\mathbb{b}_{30}^{\alpha;k,1}), \tag{6k'}$$

$$\begin{aligned}\mathbb{b}_{10}^{\alpha;k,1} := \frac{1}{-2\omega_1^k}(&6\mathbb{o}_{20}^{\alpha,k} - (6-\omega_0^k+2\omega_1^k)\mathbb{b}_{20}^{\alpha;k,1}\\ &- \omega_0^k \mathbb{b}_{30}^{\alpha;k,1}).\end{aligned} \tag{5k'}$$

(iii) If $w := \omega_1^{\mu_\ell} \neq 0$ (e.g. $\omega_1^{\mu_\ell} = 1$ for a $\langle 6,3\rangle$ edge, or $\omega_1^{\mu_\ell} = -1$ for a $\langle 3,6\rangle$ edge) then for $k := 1$

$$\begin{bmatrix}\mathbb{b}_{10}^{\alpha;k,1}\\ \mathbb{b}_{20}^{\alpha;k,1}\\ \mathbb{b}_{30}^{\alpha;k,1}\end{bmatrix} = M_w \begin{bmatrix}\mathbb{b}_{00}^{\alpha;1,1}\\ \mathbb{o}_{10}^{\alpha,1}\\ \mathbb{o}_{20}^{\alpha,1}\\ \mathbb{o}_{10}^{\alpha,2}\end{bmatrix} \tag{24}$$

$$M_1 := \frac{1}{122}\begin{bmatrix}22 & 132 & -34 & 2\\ 6 & 36 & 85 & -5\\ 2 & 12 & 69 & 39\end{bmatrix},$$

$$M_{-1} := \frac{1}{46}\begin{bmatrix}-6 & 36 & 14 & 2\\ 2 & -12 & 49 & 7\\ -2 & 12 & -3 & 39\end{bmatrix}.$$

and then for $k = 2,\ldots,\mu_\ell - 1$,

$$\mathbb{b}_{10}^{\alpha;k,1} := \frac{1}{\omega_0^k}(2\mathbb{o}_{k-1}^{\alpha} - (2-\omega_0^k)\mathbb{b}_{00}^{\alpha;k,1}), \tag{3k"}$$

$$\begin{aligned}\mathbb{b}_{20}^{\alpha;k,1} := \frac{1}{2\omega_0^k}(&6\mathbb{o}_{10}^{\alpha,k} - (6-2\omega_0^k+\omega_1^k)\mathbb{b}_{10}^{\alpha;k,1}\\ &+ \omega_1^k \mathbb{b}_{00}^{\alpha;k,1}).\end{aligned} \tag{4k"}$$

$$\mathbb{b}_{30}^{\alpha;k,1} := \frac{1}{2+\omega_1^k}(2\mathbb{o}_k^{\alpha} + \omega_1^k \mathbb{b}_{20}^{\alpha;k,1}). \tag{6k"}$$

Lemma 4. *Let $\hat{\mathbb{b}^\circ}$ be the inner Bernstein-Bézier control points output by PGS. If $\mathbb{c} = \hat{\mathbb{b}^\circ}$ then for $\mathbb{b}_{i0}^{\alpha,k,1}$, $i \in \{0,1,2,3\}$ constructed by Edge Recovery,* (3) – (6) *hold.*

Proof The equation labels in the range of $3,\ldots,6$ of Edge Recovery indicate the G^1 constraints that are enforced by solving for the boundary coefficients. Since all coefficients are determined by these constraints and since the $\hat{\mathbb{b}^\circ}$ have not been perturbed, the output of PGS has been reconstructed. Therefor also the missing equations of type (4k') and (5k") must hold. |||

4.2. Algorithm PGS^{ER}

We define an alternative algorithm PGS$^{\text{ER}}$ by replacing Step 0 of algorithm PGS with the more sophisticated Edge Recovery and then applying the remaining Steps 1-5 of PGS. This yields a surface with $\hat{\mathbb{b}^\circ} = \mathbb{c}$, i.e. $\mathbb{b}^\circ$ is preserved when running the modified algorithm PGS$^{\text{ER}}$.

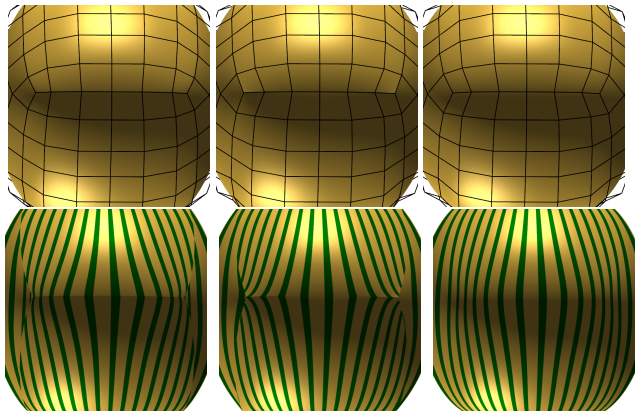

Figure 11: (Top) Surface with BB-net. (Bottom) highlight lines. (a) Initialized mesh after Step 0; (b) Edge Recovery applied to input; (c) output of PGS$^{\text{ER}}$.

Corollary 1. *Applying PGS^{ER} with $\mathbb{c} := \hat{\mathbb{b}^\circ}$ reproduces the PGS surface with inner Bézier coefficients $\hat{\mathbb{b}^\circ}$.*

We can now derive the d.o.f. $\mathbb{c}^{\ell+1}$ of the representation at level $\ell+1$ by applying de Casteljau's algorithm to the representation at level ℓ. If the $\mathbb{c}^{\ell+1}$ are not perturbed, i.e. $\mathbb{c}^{\ell+1} = \hat{\mathbb{b}^\circ}^{\ell+1}$ then we represent the surface exactly, otherwise we apply Steps 1-5 of PGS to obtain a surface that is tangent continuous except possibly a long edges $\langle 4,m\rangle$, $m \neq 4$. Fig. 12 shows more complex example surfaces generated by PGS$^{\text{ER}}$.

5. Discussion

Since $\mathbb{c} = \hat{\mathbb{b}^\circ}$ changes under a second application of PGS, PGS is not suitable for generating an exact refined representation. However, since Step 0 of PGS is simple, we use it for the initial construction but use Edge Recovery in place of Step 0 thereafter for all higher levels. Experimentally, we observed

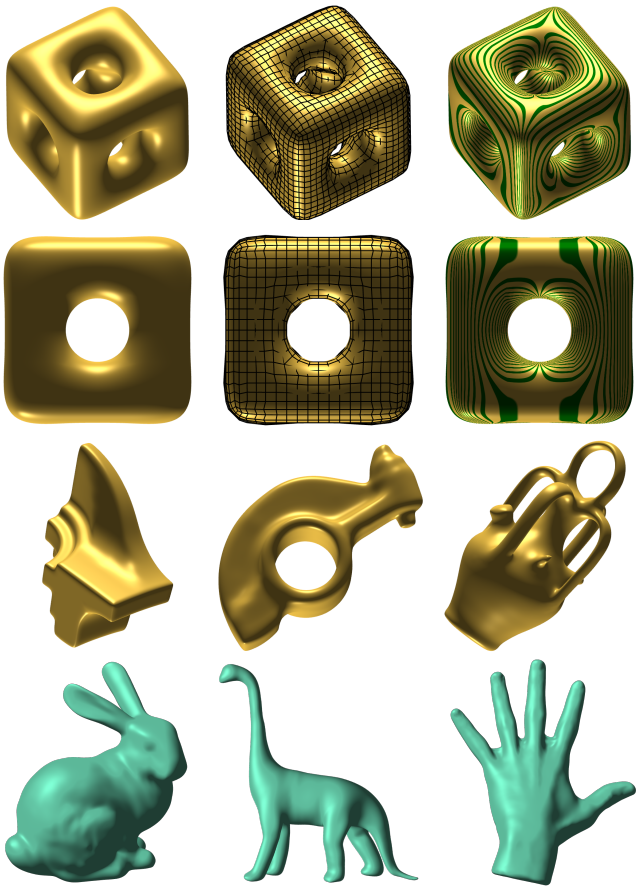

Figure 12: PGS$^{\mathrm{ER}}$ surfaces of moderate (top three rows) and high patch count (bottom row.)

that surfaces generated by PGS have slightly more uniformly distributed highlight lines than those of PGS$^{\mathrm{ER}}$ probably due to the underlying C^2 spline interpretation.

To derive a basis for each of the refined spaces, we apply PGS$^{\mathrm{ER}}$ when one $\mathbb{c}_{ij}^{\alpha;kl} = 1$ and all other $\mathbb{c}$ are zero. This yields a *B-spline-like basic function* analogous to a bi-cubic B-spline with double knots in the tensor-product case but with larger support, all along edges. (For individual basic functions $\mathbb{b}_{ij}^{\alpha;kl} \neq \mathbb{c}_{ij}^{\alpha;kl}$ but the linear combination of all basic functions with weights $\mathbb{c}$ generates a piecewise polynomial surface with $\mathbb{b}^\circ = \mathbb{c}$.)

Due to the subtle but fundamental constraints proven in [PF10], the algorithm cannot guarantee smoothness for paths with edge sequences $\langle n, 4\rangle$ and $\langle 4, m\rangle$, $n \neq m$. A well-known remedy is to choose a quadratic $\omega(t) := 2c_i(1-t)^2$, $i \in \{n, m\}$ for these edges. However, a quadratic ω for bicubic patches meeting with geometric continuity implies that the shared boundaries are piecewise quadratic rather than cubic (or we loose vector-valued degrees of freedom). If they are additionally C^2-connected due to Lemma 2 then refinement does not generate additional d.o.f. along the shared boundaries, because a C^2 piecewise quadratic curve is a single quadratic curve. Moreover, quadratic boundary curve pieces result in shape defects near higher-order saddles. Finally, varying reparameterizations makes refinement more complex. For all these reasons, we currently choose to not enforce smoothness across $\langle n, 4\rangle$ edges.

The goal of Edge Labeling is to assign outgoing labels to vertices of valence 4 and 5 to minimize the number of edge sequences with labels $\langle m, 4\rangle\ \langle 4, n\rangle$ where $m \neq n$. Edge Labeling typically succeeds in eliminating such sequences but deciding whether and how to optimize the labels or adjust the quad mesh to guarantee elimination remains an open problem.

We note that for rational linear reparameterizations, the restrictions on edge valence labels $\langle 4, n\rangle$, $n \neq 4$ hold also for surfaces of higher degree and that higher degree results in rapid growth of d.o.f. which is often undesirable. Future research may show whether better shape of higher degree constructions offsets its drawbacks.

6. Conclusion

The *local* Edge Recovery algorithm allows determining basic functions, four per input quad. These d.o.f. are uniformly distributed and correspond to the interior coefficients of bicubic Bézier patches. The refinability and nestedness of this space, also across G-boundaries, complements the treatment of T-corners in the tensor-product setting by PHT splines [KXCD15]. PGS$^{\mathrm{ER}}$ therefore yields a richer, although not fully general set of surfaces that are smooth under the simplest possible, namely rational linear reparameterization. This simplicity makes the new class of $G-$splines the closest cousin of regular tensor-product splines that are defined over the plane.

Acknowledgments.

We thank [EBCK13, ULP+15] for providing their quad mesh data. The work was supported in part by NSF Grant CCF-1117695.

[BGN14] C Beccari, D Gonsor, and M Neamtu. Rags: Rational geometric splines for surfaces of arbitrary topology. *Computer Aided Geometric Design*, 31:97–110, 2014.

[CST15] Annabelle Collin, Giancarlo Sangalli, and Thomas Takacs. Approximation properties of multi-patch C^1 isogeometric spaces. *arXiv:1509.07619*, 2015.

[EBCK13] Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. Qex: Robust quad mesh extraction. *ACM Trans. Graph.*, 32(6):168:1–168:10, November 2013.

[FB88] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. In *SIGGRAPH '88 (15th Annual Conference on Computer Graphics and Interactive Techniques, Atlanta, GA, August 1–5, 1988)*, pages 205–212, 1988.

[HWFQ09] Ying He, Hongyu Wang, Chi-Wing Fu, and Hong Qin. A divide-and-conquer approach for automatic polycube map construction. *Computers & Graphics*, 33(3):369 – 380, 2009.

[KXCD15] Hongmei Kang, Jinlan Xu, Falai Chen, and Jiansong Deng. A new basis for PHT-splines. *Graphical Models*, 82:149–159, 2015.

[LJFW08] Juncong Lin, Xiaogang Jin, Zhengwen Fan, and Charlie C. L. Wang. Automatic polycube-maps. In *Proceedings of the 5th International Conference on Advances in Geometric Modeling and Processing*, GMP'08, pages 3–16, Berlin, Heidelberg, 2008. Springer-Verlag.

[PF09] J. Peters and Jianhua Fan. On the complexity of smooth spline surfaces from quad meshes. *Computer-Aided Geometric Design*, 27:96–105, 2009.

[PF10] Jörg Peters and Jianhua Fan. The projective linear transition map for constructing smooth surfaces. In J.P.Pernot, editor, *Shape*

Modelling International, SMI10 conference, pages 124–130. IEEE Computer Society, June 21-23 2010.

[PS15] Jörg Peters and Martin Sarov. Polynomial spline surfaces with rational linear transitions. *Computers & Graphics*, 51: 43–51,2015.

[THCM04] Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. Polycube-maps. *ACM Trans. Graph.*, 23(3):853–860, August 2004.

[ULP+15] Francesco Usai, Marco Livesu, Enrico Puppo, Marco Tarini, and Riccardo Scateni. Extraction of the quad layout of a triangle mesh guided by its curve skeleton. *ACM Trans. Graph.*, 35(1):6:1–6:13, December 2015.

[WHL+08] Hongyu Wang, Ying He, Xin Li, Xianfeng Gu, and Hong Qin. Polycube splines. *Computer-Aided Design*, 40(6):721 – 733, 2008.

[WJH+08] Hongyu Wang, Miao Jin, Ying He, Xianfeng Gu, and Hong Qin. User-controllable polycube map for manifold spline construction. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, SPM '08, pages 397–404, New York, NY, USA, 2008. ACM.

[WYZ+11] Shenghua Wan, Zhao Yin, Kang Zhang, Hongchao Zhang, and Xin Li. A topology-preserving optimization algorithm for polycube mapping. *Computers & Graphics*, 35(3):639–649, June 2011.

[XGH+11] Jiazhil Xia, Ismael Garcia, Ying He, Shi-Qing Xin, and Gustavo Patow. Editable polycube map for GPU-based subdivision surfaces. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 151–158, New York, NY, USA, 2011. ACM.

A. Edge Labeling

The goal of the Edge Labeling is to minimize the number of edge sequences with labels $\langle m, 4\rangle$ $\langle 4, n\rangle$ where $m \neq n$. It is applied to the unrefined quad mesh.

Edge Labeling:

Input A (polycube) quad mesh with valences $3, 4, 5, 6$.

Output A labeling $\langle p, q\rangle$ of each edge showing the *apparent valence* of its endpoints as in (8).

We will use, for each vertex $\mathbf{v}$, an auxiliary set of n integers, called edge-vertex *markers*. There is one marker for each *outgoing label* p on the edge labeled $\langle p, q\rangle$. We proceed as follows.

1. Initialize all edge labels with the true vertex valence at either end. Initialize all markers as zero.
2. For each edge with outgoing label $\lambda = 6$ trace a path until a vertex of valence $n \neq 4$ is encountered. The 4-valent vertices along the path are traversed by always proceeding to the opposite edge, i.e. from edge e_i to e_{i+2} with subscripts modulo 4. Set the marker of the incoming edge to 1.
3. For all vertices of valence $p = 5$, if possible, assign the outgoing labels marked 1 with $\lambda = 6$ as in Fig. 5, left. For each edge with outgoing label 6, trace the path back until a label 6 is found, setting the markers of the 4-valent vertices along the path to 1.
4. For each edge with outgoing label 3 trace the path until a label 3 or 6 is found, setting the markers of all 4-valent vertices along the path to 1 for incoming edges and -1 for outgoing edges.
5. For each vertex $\mathbf{v}_p$ of valence $p = 4$ such that all four markers are nonzero (i.e. two paths cross), collect the labels λ_i and λ_{i+2} of opposing neighbors i and $i + 2$, $i = 0, 1$. If $\lambda_i = \lambda_{i+2}$ then label edge i as $\langle \lambda_i, 4\rangle$ and edge $i + 2$ as $\langle 4, \lambda_i\rangle$. Otherwise label edge i as $\langle \lambda_i, \lambda_{i+2}\rangle$ and label edge $i + 2$ as $\langle \lambda_{i+2}, \lambda_i\rangle$. (This yields, for example, a path from a vertex with valence 6 to one with valence 3 such that every segment has the same label-pair $\langle 6, 3\rangle$.)
   (a) If all four labels corresponding to $\mathbf{v}_p$ are either 6 or 3, then overwrite them by 4. (This will yield four C^0 edges).
   (b) For vertices with 3 or 1 neighbors' labels set, set the unmatched neighbor label to that of its opposite: $\lambda_i = \lambda_{i+2}$ and the corresponding two labels of $\mathbf{v}_p$ to 4. (This yields a sequence $\langle m, 4\rangle, \langle 4, m\rangle$ and hence a G^1 construction.)
   (c) For all vertices that have 2 or 0 of their neighbors' labels set, set the remaining neighbors' labels to 6 and the corresponding labels of $\mathbf{v}_p$ to 4. (This yields a sequence $\langle 6, 4\rangle, \langle 4, 6\rangle$ and hence a G^1 construction.)
6. For each vertex $\mathbf{v}_p$ of valence $p = 4$ such that only two markers are non-zero (i.e. a single path), set the label to 6 where the mark is 1 and to 3 if the mark is -1. (This yields a path with G^1 edges $\langle 3, 6\rangle$.)