DOI:10.1111/cgf.14074
Eurographics Symposium on Geometry Processing 2020
Q. Huang and A. Jacobson
(Guest Editors)

Volume 39 (2020), Number 5

Hexahedral Mesh Repair via Sum-of-Squares Relaxation

Z. Marschner, D. Palmer, P. Zhang, and J. Solomon

Massachusetts Institute of Technology, USA

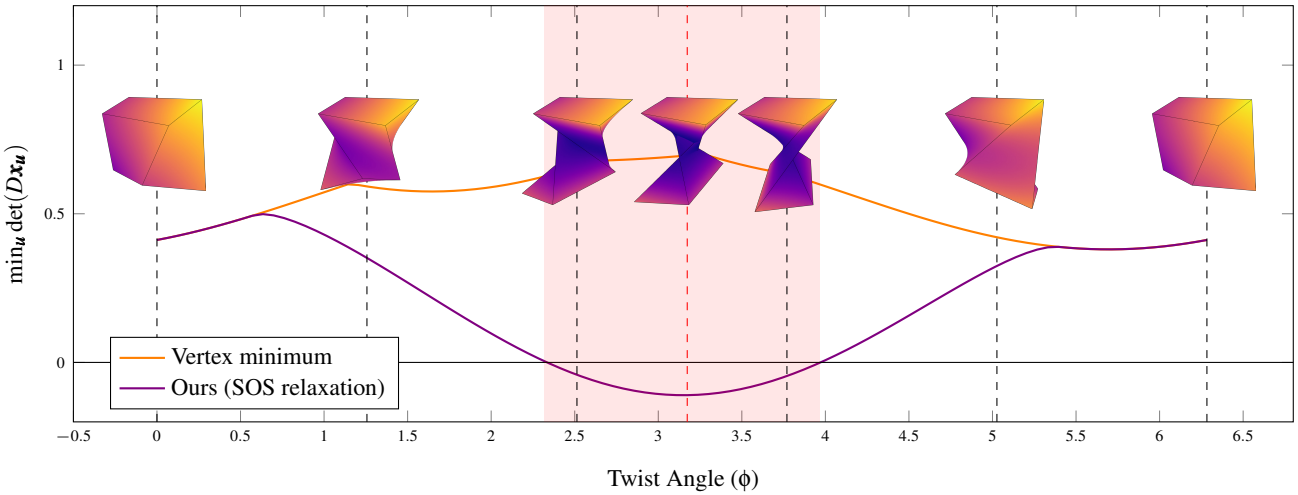

Figure 1: *In this illustrative example, we plot the Jacobian determinant over a hex as its bottom four vertices are twisted about the vertical axis. The minimum Jacobian among the eight vertices vastly overestimates the minimum Jacobian in the interior of the hex, failing to capture the invalidity which occurs in the red range of angles. Our SOS relaxation, in contrast, computes the true minimum.*

Abstract

The validity of trilinear hexahedral (hex) mesh elements is a prerequisite for many applications of hex meshes, such as finite element analysis. A commonly used check for hex mesh validity evaluates mesh quality on the corners of the parameter domain of each hex, an insufficient condition that neglects invalidity elsewhere in the element, but is straightforward to compute. Hex mesh quality optimizations using this validity criterion suffer by being unable to detect invalidities in a hex mesh reliably, let alone fix them. We rectify these challenges by leveraging sum-of-squares relaxations to pinpoint invalidities in a hex mesh efficiently and robustly. Furthermore, we design a hex mesh repair algorithm that can certify validity of the entire hex mesh. We demonstrate our hex mesh repair algorithm on a dataset of meshes that include hexes with both corner and face-interior invalidities and demonstrate that where naïve algorithms would fail to even detect invalidities, we are able to repair them. Our novel methodology also introduces the general machinery of sum-of-squares relaxation to geometry processing, where it has the potential to solve related problems.

1. Introduction

Hexahedral meshes have been shown to have superior numerical properties to tetrahedral meshes for solving various numerical PDEs. This is especially true when comparing trilinear hexahedra to linear tetrahedra [CK92, Wei94], but also extends to quadratic bases in nonlinear elasto-plastic simulation [BPM*95]. Motivated by enhanced performance in simulation, significant effort has been devoted toward automatic generation of high-quality hexahedral meshes [NRP11]. These algorithms use a variety of constructions and mathematical approaches, from frame fields [HTWB11, SVB17, PBS20] to polycubes [FXBH16, THCM04, HJS*14, HZ16], to octrees [QZ12], to topology constraints [FM99, LZC*18, CC19].

Many metrics are used to evaluate hex mesh validity and quality. The Jacobian determinant of the trilinear map from a regular unit cube to the hex element is a frequently-used validity metric whose positivity guarantees injectivity of the map [Knu03]. An element's validity (and sometimes quality) is typically assessed by the minimum of the determinants at the corners, optionally normalized by the edge lengths [XGC17, GPW*17]. Positivity of this summary metric, however, is insufficient to guarantee positivity of the Jacobian determinant everywhere in the interior of the hex element [Knu90]. The degree to which vertex-based verification of hex validity is insufficient is demonstrated in Figure 2. We visualize a large region in the space of hexes where the hex is invalid, but passes the vertex test. It is conjectured that having positive Jacobian determinant on all six bi-quadratic faces of a hex element is sufficient for positivity of the determinant on the interior [Knu90], however this condition remains unproven and is not used in practice.

In this paper, we present a new perspective on the problem of checking hex mesh validity. In contrast to previous work, which relies on an iterative refinement strategy to localize points of invalidity, we express the validity problem over the entire hex as a *polynomial optimization problem* and solve it via the machinery of *sum-of-squares* (SOS) programming. This makes for a simple-to-implement validity check that produces a certificate of validity up to numerical precision. Using the dual *moment relaxation*, we obtain the point at which the Jacobian determinant is minimized; this can then be used as a primitive in a simple mesh repair algorithm. In summary, we

- formulate an SOS relaxation that computes the minimum Jacobian determinant value inside a trilinear hex element;
- formulate a moment relaxation that computes the location of the minimum Jacobian determinant; and
- using these new relaxations, design a global hex repair algorithm whose success guarantees injectivity of the mesh elements.

2. Related Work

2.1. Hexahedral Mesh Validity

A full review of hex mesh evaluation methods is out of scope for this paper. For an extensive report on the most popular hex quality metrics, see [SEK*07]. Many of these metrics are interactively viewable on a database of hex meshes via [BPLC19]. A correlation-based evaluation of these different metrics is presented in [GHX*17]. We will focus on metrics related to injectivity of the trilinear map, i.e., those that use the Jacobian determinant.

A variety of methods propose heuristics for verifying positivity of the Jacobian determinant by evaluating volumes of subtetrahedra [Gra99, Ush01, Vav03, Zha05]. The volume of the hexahedron itself is also sometimes used [Ush01]. These metrics were studied empirically in [Ush11] and found to be insufficient.

Past work on verifying positivity of the Jacobian determinant over an entire hex relies on iterative refinement, i.e., branch-and-bound. [DOS99] points out that the property that a Bézier function lies in the convex hull of its control points allows one to bound the Jacobian determinant. [HMESM06] applies this property in combination with iterative refinement to check injectivity of triangular

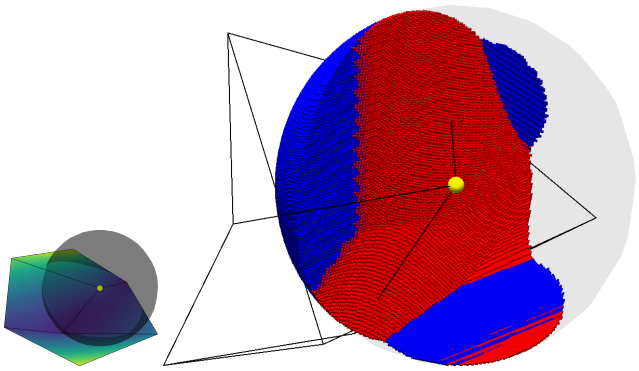

Figure 2: *Starting from the valid hex outlined with black edges, we displace the yellow vertex within a ball and color the positions that make the hex invalid. In the blue region, the hex has a corner inversion. In the red region, the hex has a non-corner inversion, and checking the Jacobian determinant at corners is insufficient to detect its invalidity.*

Bézier patches. [JRG12] develops a branch-and-bound approach to computing validity of curvilinear elements. [JWR17] leverages specific properties of hexahedral elements to improve efficiency significantly. By recursively subdividing, the method progressively tightens bounds on the minimal Jacobian determinant, relying on [Ler08, Ler09] to prove that this subdivision scheme terminates.

2.2. Hexahedral Mesh Repair

Mesh quality improvement is a long-studied topic with a variety of heuristics ranging from smoothing [Knu99, ZBX09, QZW*10, HZX18] to boundary relaxation [RGRS14] to extending tetrahedral mesh quality improvement methods to hex meshes [Knu00, WSR*12]. [LXZQ13] combines surface fairing and pillowing to improve hex mesh quality. [LSVT15] measures the quality of a hex mesh from the volumes of tetrahedra surrounding its directed edges and use that information to derive a mesh quality optimization strategy. [XGC17] explicitly fixes regions of negative Jacobian determinant by applying boundary relaxations, untangling procedures, and a non-inverting volume deformation. This method, however, only computes Jacobian determinants at vertices. As we mention above, approaches using only vertex based inversion detection cannot detect all inverted hex elements, let alone correct them. [BDK*03] aggregates many mesh quality improvement methods into the Mesquite library. Of the hex mesh repair algorithms mentioned above, none guarantees validity of the final mesh—even on the condition that the algorithm terminates. [TGRL13] untangles curvilinear meshes by optimizing conservative bounds on the Jacobian determinant within each element (relying on the Bézier basis–convex hull property referenced in § 2.1). This method can guarantee validity of mesh elements, but does not compute the minimum Jacobian or its location.

2.3. Sum-of-squares and Semidefinite Relaxation

Sum-of-squares (SOS) relaxation is a general technique for approximating solutions to polynomial optimization problems—that is,

optimization problems whose objective and constraint functions are polynomials in the variables. A comprehensive review of this field is presented in [BPT12]. We present mathematical constructions relevant to this paper in § 3.

SOS relaxation is an example of a general technique called convex relaxation, in which an optimization problem is extended from a nonconvex domain to a larger convex one with the hope that the global optimum to the convex problem will yield a global optimum of the original problem. Among relaxations, SOS relaxations are expecially practical because they yield semidefinite optimization problems (SDP), which are solvable in polynomial time via interior-point methods [Ali95, NN94, BBV04] or rank-deficient first-order methods [BM03, BVB18, CM19]. The simplest example of SOS relaxation transforms quadratically constrained quadratic programs (QCQP) to SDP. Perhaps the most famous computational application of QCQP–SDP relaxation is the approximation algorithm for MAX-CUT in [GW95]. Graphics and geometry processing research has employed semidefinite relaxation to solve problems as varied as rotation synchronization [Sin11], point cloud registration [MDK*16], consistent mapping [HG13], point cloud encapsulation [AHMS17], and even frame field optimization for hex meshing [PBS20].

QCQP–SDP relaxations can be viewed as the first level of a hierarchy of ever-richer SOS relaxations yielding ever-larger SDPs. This is known as the *Lasserre hierarchy* [Las01] of moment relaxations. Higher-degree SOS relaxations have many computational applications, including robust optimization [BTEGN09], robust control [ZDG96], Lyapunov stability [Par00, PP02], and quantum separability [BCY11], among others detailed in §3 of [BPT12]. To the authors' knowledge, this paper is among the first in computer graphics to use the full SOS relaxation hierarchy.

3. Preliminaries

The sum-of-squares theory is vast and rich in connections to algebraic geometry, combinatorics, and complexity theory. For completeness, we introduce the notions relevant to our work here and refer the reader to [BPT12] for a deeper introduction.

3.1. Notation

The ring of real polynomials on n variables is $\mathbb{R}[\boldsymbol{x}] = \mathbb{R}[x_1, \dots, x_n]$. $\mathbb{R}[\boldsymbol{x}]_d$ will denote polynomials of degree at most d. We will sometimes write a polynomial of degree d in the compact form

$$p(\boldsymbol{x}) = \sum_{|\alpha| \le d} p_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}},$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$ is a **multi-index**, i.e., $\boldsymbol{x}^{\boldsymbol{\alpha}} := \prod_i x_i^{\alpha_i}$ and $|\boldsymbol{\alpha}| := \sum_i \alpha_i$.

3.2. SOS Polynomials

The simplest polynomial optimization problem asks whether a given polynomial is nonnegative on all of $\mathbb{R}^n$. The set of nonnegative polynomials of (even) degree $2d$ in n variables is denoted $P_{n,2d}$. Geometrically, $P_{n,2d}$ is a proper cone in $\mathbb{R}[x_1, \dots, x_n]_{2d}$. How can we decide membership in this set? An algorithmic answer to this question begins with the observation that the square of any polynomial will evaluate to a nonnegative number:

Definition 3.1 (SOS Polynomials)**.** A real polynomial $q(x_1, \dots, x_n) \in \mathbb{R}[x_1, \dots, x_n]$ of even degree is a **sum-of-squares (SOS) polynomial** if it can be written as

$$q(\mathbf{x}) = \sum_i q_i(\mathbf{x})^2 \tag{1}$$

for some polynomials q_i for $i = 1$ to n and $\mathbf{x} = (x_1, \dots, x_n)$.

The set of SOS polynomials of degree $2d$ in n variables is denoted $\Sigma_{n,2d}$. We will write $\Sigma_n = \bigcup_d \Sigma_{n,2d}$. Since any SOS polynomial is nonnegative, $\Sigma_{n,2d} \subseteq P_{n,2d}$.

While deciding positivity—membership in $P_{n,2d}$—is NP-hard in general (see [BPT12] §3.4.3, [GV01]), deciding membership in $\Sigma_{n,2d}$ amounts to an SDP. In particular, rewriting (1) makes this equivalence explicit:

$$\begin{aligned} q(\mathbf{x}) &= \begin{pmatrix} q_1(\mathbf{x}) \\ \vdots \\ q_m(\mathbf{x}) \end{pmatrix}^\top \begin{pmatrix} q_1(\mathbf{x}) \\ \vdots \\ q_m(\mathbf{x}) \end{pmatrix} = [\boldsymbol{x}]_d^\top \underbrace{\begin{pmatrix} \boldsymbol{q}_1 \\ \vdots \\ \boldsymbol{q}_m \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{q}_1 \\ \vdots \\ \boldsymbol{q}_m \end{pmatrix}}_{Q} [\boldsymbol{x}]_d \\ &= \left\langle Q, [\boldsymbol{x}]_d [\boldsymbol{x}]_d^\top \right\rangle, \end{aligned} \tag{2}$$

where $[\boldsymbol{x}]_d$ is the basis of monomials of degree up to d, $\boldsymbol{q}_i$ is the vector of coefficients of q_i in this basis, and $\langle , \rangle$ denotes the Frobenius inner product. (2) says that the coefficients of a SOS polynomial q can be written in terms of a positive semidefinite matrix Q.

3.3. SOS on a Compact Domain

The central theorem in the theory of SOS programming is the **Positivstellensatz**. Given any polynomial optimization problem on a domain defined by polynomial equality and inequality constraints, it guarantees that the global solution can be computed by an SOS relaxation of sufficiently high degree—i.e., by a sufficiently large SDP. Because our polynomial optimization problem is over the unit cube, we use Putinar's simpler variant of the Positivstellensatz for compact domains [Put93].

Definition 3.2 (Nonnegative Locus)**.** Given a set of real polynomials $G = \{g_1, \dots, g_m\} \subset \mathbb{R}[x_1, \dots, x_n]$, the **nonnegative locus** of G is the set of points on which all the polynomials in G are nonnegative:

$$P(G) = \{\boldsymbol{x} \in \mathbb{R}^n : g_1(\boldsymbol{x}) \ge 0, \dots, g_m(\boldsymbol{x}) \ge 0\}. \tag{3}$$

Definition 3.3 (Quadratic Module)**.** Given $G = \{g_1, \dots, g_m\} \subset \mathbb{R}[x_1, \dots, x_n]$, the **quadratic module** of G is

$$Q(G) = \{p(\boldsymbol{x}) : p(\boldsymbol{x}) = s_0(\boldsymbol{x}) + \sum_{i=1}^m s_i(\boldsymbol{x}) g_i(\boldsymbol{x}), \quad s_i \in \Sigma_n\}. \tag{4}$$

We will use $Q_{2d}(G)$ to denote the slice of $Q(G)$ consisting of polynomials p that admit a decomposition with $\deg s_0 \le 2d$ and $\deg s_i \le 2d - \deg g_i$ for each $i \in \{1, \dots, m\}$.

It is clear that any polynomial in $Q(G)$ is nonnegative on $P(G)$. One might hope that the converse is true—that any polynomial which is positive on $P(G)$ could be written in the special form (4). Putinar's theorem states exactly this when $Q(G)$ is *Archimedean*.

Definition 3.4 (Archimedean)**.** A quadratic module $Q(G)$ is **Archimedean** if there exists a polynomial $q(\boldsymbol{x}) \in Q(G)$ such that $P(\{q\})$ is compact. In this case, we have $P(G) \subseteq P(\{q\})$ also compact, and so q serves as an algebraic *certificate* of the compactness of $P(G)$. For the purposes of this paper, we will also refer to $P(G)$ as an **Archimedean set**.

Once $Q(G)$ is verified to be Archimedean, one can apply Theorem 3.1.

Theorem 3.1 (Putinar's Positivstellensatz [Put93]; see also [BPT12], Theorem 3.138)**.** *Let $S = P(G) = P(\{g_1, \ldots, g_m\})$ be an Archimedean set defined by the polynomial inequalities $g_i(\boldsymbol{x}) \geq 0$. Then* any *polynomial $p(\boldsymbol{x})$ that is strictly positive on S is in $Q(G)$, i.e., there exists a decomposition*

$$p(\boldsymbol{x}) = s_0(\boldsymbol{x}) + \sum_{i=1}^{m} s_i(\boldsymbol{x}) g_i(\boldsymbol{x}), \tag{5}$$

with SOS polynomials $s_i \in \Sigma_{n,2d}$, for high enough degree d.

3.4. Moment Relaxation

Suppose we want to find the global minimum of a polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ on a domain S. This problem is nonconvex, but it can be rewritten as a convex problem by the following **measure relaxation**, in which we replace evaluation at a point by integration against an arbitrary probability measure:

$$p_S^* \coloneqq \min_{\mu \in \mathcal{P}(S)} \mathbf{E}_\mu[p], \tag{6}$$

where $\mathcal{P}(S)$ is the space of probability measures on S, and $\mathbf{E}_\mu[p] = \int_S p \, d\mu$ denotes integration against μ. Indeed, if $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in S} p(\boldsymbol{x})$, then the atomic measure $\delta_{\boldsymbol{x}^*}$ minimizes (6) [Las01]. That is, the measure relaxation (6) is exact.

Problem (6) optimizes over the infinite-dimensional space $\mathcal{P}(S)$, making it intractable to solve directly. However, the objective function can be re-written to only depend on a finite set of real numbers computed from μ, namely **moments** of μ of degree $\leq d$:

$$\mathbf{E}_\mu[p] = \sum_{|\alpha| \leq d} p_{\boldsymbol{\alpha}} \mathbf{E}_\mu[\boldsymbol{x}^{\boldsymbol{\alpha}}] = \sum_{|\alpha| \leq d} p_{\boldsymbol{\alpha}} \mu_{\boldsymbol{\alpha}}, \tag{7}$$

where $d = \deg p$ and $\mu_{\boldsymbol{\alpha}} \coloneqq \mathbf{E}_\mu[\boldsymbol{x}^{\boldsymbol{\alpha}}]$ is the $\boldsymbol{\alpha}$-moment of μ.

Notation. We will write $\mathcal{M}_d$ for the vector space of moment vectors of degree up to d and $\boldsymbol{\mu} = \boldsymbol{M}(\mu) \in \mathcal{M}_{2d}$ for the vector of moments of μ up to degree $2d$. (7) gives a way to write expectations of polynomials with respect to moment vectors without recourse to measures; this is called **pseudo-expectation** and is written $\mathbf{E}_{\boldsymbol{\mu}}$. We may also index moments by monomials instead of multi-index $\boldsymbol{\alpha}$ e.g. $\mu_{(1,1,0)} = \mu_{x_1 x_2}$.

If it were possible to express the constraint $\mu \in \mathcal{P}(S)$ in terms of moments, we could reduce (6) to a finite-dimensional problem. In general, an arbitrary set of moments may not correspond to any probability measure on S. Happily, when S is Archimedean, the following theorem of Lasserre tells us that a finite number of constraints is sufficient to characterize the feasible set of moments for a polynomial optimization of the form (6):

Theorem 3.2 ([Las01], Theorem 4.2)**.** *Suppose $S = P(G)$ is Archimedean. Let $p(\boldsymbol{x})$ be a polynomial, and let $p_S^* \coloneqq \min_{\boldsymbol{x} \in S} p(\boldsymbol{x})$ be its minimum on S, occurring at $\boldsymbol{x}^* \in S$. Let d be large enough*

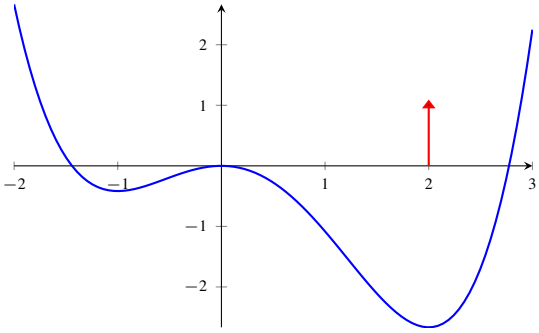

Figure 3: *The optimal solution to the moment relaxation (8) is the set of moments of an atomic measure supported at the global minimum of the polynomial on the domain—in this case, δ_2.*

that $p(\boldsymbol{x}) - p_S^* \in Q_{2d}(G)$, which must exist by Theorem 3.1. Then the following ***moment relaxation*** *computes p_S^*:*

$$p_S^* = \left\{ \begin{array}{lll} \min\limits_{\boldsymbol{\mu} \in \mathcal{M}_{2d}} & \mathbf{E}_{\boldsymbol{\mu}}[p(\boldsymbol{x})] & \\ \text{s.t.} & \mathbf{E}_{\boldsymbol{\mu}}[q(\boldsymbol{x})^2] \geq 0, & \forall q \in \mathbb{R}[\boldsymbol{x}]_d \\ & \mathbf{E}_{\boldsymbol{\mu}}[q(\boldsymbol{x})^2 g_i(\boldsymbol{x})] \geq 0, & \forall q \in \mathbb{R}[\boldsymbol{x}]_{d - w_i} \\ & \mathbf{E}_{\boldsymbol{\mu}}[1] = 1, & \end{array} \right\} \tag{8}$$

where $w_i = \lceil \deg g_i / 2 \rceil$. Moreover, the moment vector $\boldsymbol{\delta}^* \coloneqq \boldsymbol{M}(\delta_{\boldsymbol{x}^*})$ is a global minimizer of (8).

It is worth examining Theorem 3.2 more closely. First, the relaxation (8) is an SDP. Each inequality constraint can be written as a semidefinite constraint on a **moment matrix**. For example, the first constraint in (8) is equivalent to $\mathbf{E}_{\boldsymbol{\mu}}[[\boldsymbol{x}]_d [\boldsymbol{x}]_d^\top] \succeq 0$. In the univariate case ($n = 1$), this is

$$\begin{pmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots & \mu_d \\ \mu_1 & \mu_2 & \mu_3 & \cdots & \mu_{d+1} \\ \mu_2 & \mu_3 & \mu_4 & \cdots & \mu_{d+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_d & \mu_{d+1} & \mu_{d+2} & \cdots & \mu_{2d} \end{pmatrix} \succeq 0. \tag{9}$$

Second, the relationship between Theorem 3.1 and Theorem 3.2 is precisely that of SDP duality [Las01]. This is why the same degree d appears in both statements.

Third, as with Theorem 3.1, Theorem 3.2 says nothing about how high the degree d may be. Indeed, as NP-hard problems may be phrased as polynomial optimization problems, we should expect that d may be up to exponentially large in the degrees of the objective p and constraints g_i [GV01]. Even for a specific problem, it is often hard to compute d in advance. When a large enough d is chosen so that (8) holds, we say that the relaxation is **exact** or has achieved **exact recovery.**

In case of exact recovery and when the minimizer is unique, the minimizer can be computed directly from the zeroth and first moments—as the mean of the (atomic) measure. Moreover, exact recovery can be verified from the rank of the moment matrix

$$\mathbf{E}_{\boldsymbol{\mu}}[[\boldsymbol{x}]_d [\boldsymbol{x}]_d^\top] = [\boldsymbol{x}^*]_d [\boldsymbol{x}^*]_d^\top \iff \operatorname{rank} \mathbf{E}_{\boldsymbol{\mu}}[[\boldsymbol{x}]_d [\boldsymbol{x}]_d^\top] = 1. \tag{10}$$

We will use moment relaxation in § 4.3 to obtain the exact location of the most distorted point in a hex element.

4. Detecting Invalid Hexahedral Elements

In this section, we introduce the machinery of SOS relaxation to the problem of validating hexahedral mesh elements. This machinery not only certifies elements as valid or invalid, but also finds the most distorted point in each element, enabling an iterative mesh repair algorithm in the following section.

4.1. Trilinear Hexahedral Element Quality

Definition 4.1 (Trilinear Hexahedron)**.** A **trilinear hexahedron** is specified by a map $\boldsymbol{x}\colon [0,1]^3 \to \mathbb{R}^3$ that is linear in each coordinate when the others are held fixed, i.e.,

$$\boldsymbol{x}(\lambda u_0 + (1-\lambda)u_1, v, w) = \lambda \boldsymbol{x}(u_0, v, w) + (1-\lambda)\boldsymbol{x}(u_0, v, w). \quad (11)$$

Such a map admits a decomposition

$$\boldsymbol{x}(u,v,w) = \sum_{i,j,k=0}^{1} \boldsymbol{x}(i,j,k) L_{ijk}(u,v,w), \quad (12)$$

where L_{ijk} are the real-valued **Lagrange interpolation functions**

$$L_{ijk}(u,v,w) := u^i(1-u)^{1-i} v^j (1-v)^{1-j} w^k (1-w)^{1-k}. \quad (13)$$

In other words, the trilinear hexahedron is completely characterized by the positions of its vertices $\boldsymbol{x}_{ijk} = \boldsymbol{x}(i,j,k) \in \mathbb{R}^3$.

The map $\boldsymbol{x}$ is locally injective at $\boldsymbol{u} = (u,v,w)$ if its Jacobian determinant is positive, i.e.,

$$\det(D\boldsymbol{x}_{\boldsymbol{u}}) > 0. \quad (14)$$

Note that $\det(D\boldsymbol{x}_{\boldsymbol{u}})$ is a polynomial of degree 5 in three variables, where each monomial is at most quadratic in u, v, or w. Figure 1 features density plots of the Jacobian determinant for a sequence of trilinear hexahedra.

Definition 4.2 (Valid Hexahedron)**.** A hexahedron $\boldsymbol{x}$ is **valid** if it is injective everywhere, i.e.,

$$\det(D\boldsymbol{x}_{\boldsymbol{u}}) \;>\; 0, \;\; \forall \boldsymbol{u} \in [0,1]^3. \quad (15)$$

We can relate the definition of hex validity to the language of § 3 by recognizing that checking validity of a hexahedron is a *polynomial feasibility problem*. In particular, a hexahedron is valid if the solution to the following *polynomial optimization problem*, which seeks a bound on the determinant in (14), is positive:

$$\lambda^* := \left\{ \begin{array}{ll} \max\limits_{\lambda} & \lambda \\ \text{s.t.} & \det(D\boldsymbol{x}_{\boldsymbol{u}}) \geq \lambda, \quad \forall \boldsymbol{u} \in [0,1]^3 \end{array} \right\}. \quad \text{(POP)}$$

We will see in § 4.2 that through an SOS relaxation we can transform (POP) into an SDP.

4.2. Applying the SOS Relaxation

We begin with the polynomial optimization formulation of the minimal Jacobian determinant problem as described by (POP). To reformulate this as an SOS optimization problem, we seek to apply Theorem 3.1. First, we construct the set of real polynomials $G = \{g_1, \ldots, g_6\}$ whose nonnegative locus is the unit cube $P(G) = [0,1]^3$:

$$\begin{aligned} g_1(u,v,w) &= u, & g_2(u,v,w) &= 1-u, \\ g_3(u,v,w) &= v, & g_4(u,v,w) &= 1-v, \\ g_5(u,v,w) &= w, & g_6(u,v,w) &= 1-w. \end{aligned} \quad (16)$$

To use Theorem 3.1, we must verify that $Q(G)$ is Archimedean, i.e., satisfies Definition 3.4. We choose $q(\boldsymbol{x}) = N - \sum_i x_i^2$, a polynomial with compact nonnegative locus, to be our algebraic certificate of compactness. It remains to verify that $q(\boldsymbol{x}) \in Q(G)$. Choosing the SOS multipliers

$$\begin{aligned} s_1(u,v,w) &= (u-1)^2, & s_2(u,v,w) &= u^2+1, \\ s_3(u,v,w) &= (v-1)^2, & s_4(u,v,w) &= v^2+1, \\ s_5(u,v,w) &= (w-1)^2, & s_6(u,v,w) &= w^2+1, \end{aligned} \quad (17)$$

we obtain,

$$3 - u^2 - v^2 - w^2 = \sum_i s_i(u,v,w) g_i(u,v,w), \quad (18)$$

as required , with the choice $N = 3$. Then, by Equation (5), we have $\det(D\boldsymbol{x}_{\boldsymbol{u}}) \geq \lambda$ for all $\forall \boldsymbol{u} \in P(G)$ if and only if

$$\det(D\boldsymbol{x}_{\boldsymbol{u}}) - \lambda \in Q(G). \quad (19)$$

We use this to rewrite (POP) as an SOS optimization:

$$\lambda^* = \left\{ \begin{array}{ll} \max\limits_{\lambda, s_i} & \lambda \\ \text{s.t.} & \det(D\boldsymbol{x}_{\boldsymbol{u}}) - \lambda = s_0(\boldsymbol{u}) + \sum\limits_i s_i(\boldsymbol{u}) g_i(\boldsymbol{u}) \\ & s_0, \ldots, s_6 \in \Sigma_{3,2d}. \end{array} \right\} \quad \text{(SOSP)}$$

By Theorem 3.1, for a sufficiently high choice of degree $2d$,† (POP) and (SOSP) have equivalent solutions. As mentioned in § 3.2, the seven SOS polynomial constraints are encoded by corresponding coefficient matrices C_i via (2), making (SOSP) an SDP.

4.3. Obtaining the Most Distorted Point

To find the point $\boldsymbol{u}^* \in [0,1]^3$ at which the determinant is minimized, we can apply moment relaxation (see § 3.4).

$$\lambda^* = \left\{ \begin{array}{lll} \min\limits_{\boldsymbol{\mu} \in \mathcal{M}_{2d}} & \mathbf{E}_{\boldsymbol{\mu}}[\det(D\boldsymbol{x}_{\boldsymbol{u}})] & \\ \text{s.t.} & \mathbf{E}_{\boldsymbol{\mu}}[q(\boldsymbol{u})^2] \geq 0, & \forall q \in \mathbb{R}[\boldsymbol{u}]_d \\ & \mathbf{E}_{\boldsymbol{\mu}}[q(\boldsymbol{u})^2 g_i(\boldsymbol{u})] \geq 0, & \forall q \in \mathbb{R}[\boldsymbol{u}]_d \\ & \mathbf{E}_{\boldsymbol{\mu}}[1] = 1. & \end{array} \right\} \quad \text{(SOSD)}$$

† $s_1, \ldots, s_6$ actually only need to have degree $2d-1$, but since SOS polynomials always have even degree, this is increased to $2d$.

The resulting optimization problem is exactly the SDP dual to (SOSP). By Theorem 3.2, when d is sufficiently high, exact recovery occurs.

Recall from § 3.4 that when the moment relaxation is exact, the optimal moment vector $\boldsymbol{\mu}^*$ is generically that of an atomic distribution $\boldsymbol{M}(\delta_{\boldsymbol{u}^*})$. We can check for exact recovery empirically by testing the (numerical) rank of the moment matrix $\mathbf{E}_{\boldsymbol{\mu}^*}[[\boldsymbol{u}]_d[\boldsymbol{u}]_d^\top]$ corresponding to $\boldsymbol{\mu}^*$, which should be one by (10). After verifying exact recovery, we extract $\boldsymbol{u}^*$ by computing the mean of the atomic distribution directly from its first moments (see Figure 4):

$$\boldsymbol{u}^* = (\mu_u^*, \mu_v^*, \mu_w^*). \tag{20}$$

Using the rank condition, it is possible to assay empirically the degree d at which exact recovery first occurs. Our results (see Figure 5) indicate that exact recovery occurs at the lowest possible degree, $2d = 4$;‡ recall that the degree of $\det(D\boldsymbol{x}_{\boldsymbol{u}})$ is 5. Indeed, the second-largest eigenvalue of the moment matrix is zero up to numerical error, providing strong empirical evidence of exact recovery (Figure 5). Based on this data, we conjecture that $2d = 4$ is sufficient for exact recovery for any SOS program of the form (SOSP), i.e., for a hex with arbitrary vertex positions. It is remarkable that such a low degree is sufficient in practice to detect hex invalidities—while any polynomial optimization problem admits an SOS relaxtion, exact recovery is not generally guaranteed for any fixed degree. This fortuitous low degree exact recovery means that (SOSP) and (SOSD) are small SDPs, enabling our efficient mesh repair algorithm in the following section.

4.4. Implementation

We solve optimization problems (SOSP) and (SOSD) using Yalmip [Löf04] for SOS modeling and Mosek 9 [ApS19] to solve the resulting SDPs. All results are computed on a 2019 MacBook Pro with a quad-core 2.8 GHz Intel Core i7 and 16 GB RAM.

5. Hex Repair

In the previous section, we saw how SOS relaxations may be used to check validity of hex elements and find where they are invalid. Applied to a hex mesh, we can detect when individual elements are invalid (see Figure 6). We now construct a simple mesh repair algorithm built on this machinery.

5.1. Hex Element Repair

A distinguishing aspect of our SOS formulation is that it gives us the point at which the minimum Jacobian determinant is realized in a hex, rather than just coarse bounds. We can leverage this information to design a simple algorithm that transforms invalid hexes into nearby valid ones.

‡ Our relaxation differs slightly from the usual Lasserre hierarchy [Las01] by having degree $2d$ for all constraints. See previous footnote.

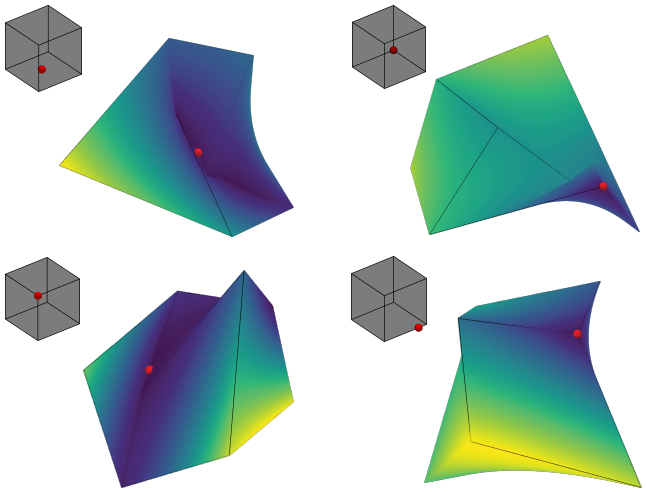

Figure 4: *For four select hexes, we show the locations of their minimum Jacobian determinants via (SOSD). The scalar Jacobian determinant field is plotted as a color map over the hex element $\boldsymbol{x}$ with yellow indicating a high value and dark blue indicating a low value, with the minimum Jacobian determinant $\boldsymbol{x}(\boldsymbol{u}^*)$ plotted as a red point. In the insets we show the location of the minimum Jacobian determinant on the parameter cube $[0,1]^3$ as a red point.*

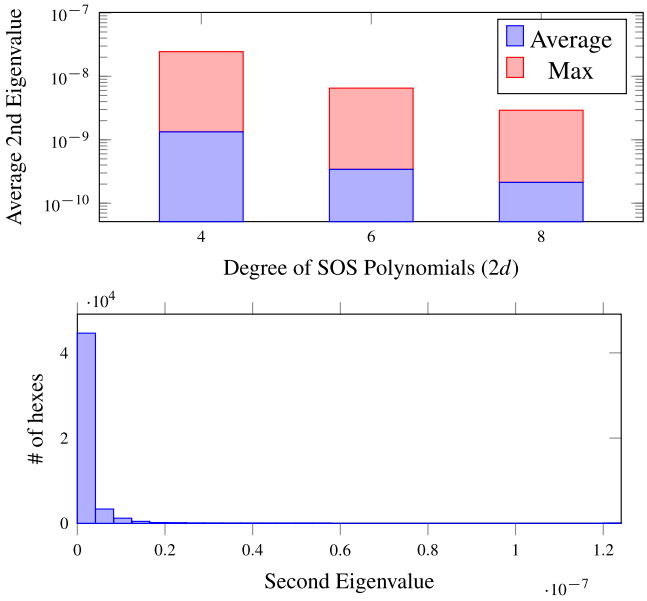

Figure 5: *(Top) The average and max second eigenvalue of the moment matrix $\mathbf{E}_{\boldsymbol{\mu}^*}[[\boldsymbol{u}]_d[\boldsymbol{u}]_d^\top]$, as explained in § 4.3. We sample 1000 hexes and solve (SOSP) with polynomials $s_1, \ldots, s_6$ taken to be varying even degrees from 4 to 8. Based on these results, we conclude that $2d = 4$ is sufficient for exact recovery of the solution. (Bottom) For degree fixed at $2d = 4$, we randomly generate 50000 hexes, solve (SOSP) and verify that their second eigenvalues are zero to numerical precision.*

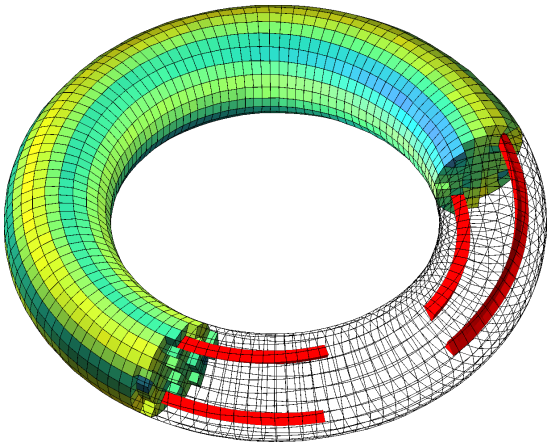

Figure 6: *The* ***torus*** *mesh from [CC19], accessed via [BPLC19]. The mesh is colored based on the minimum Jacobian determinant over that element, and in the cutaway every hex where* $\min \det(D\boldsymbol{x}_{(u,v,w)}) \leq 0$ *is highlighted in red.*

Let X be the 8×3 matrix

$$X = \begin{bmatrix} \boldsymbol{x}_{(0,0,0)} \\ \vdots \\ \boldsymbol{x}_{(1,1,1)} \end{bmatrix}$$

of vertex positions of a single hex in lexicographic order—e.g., $\boldsymbol{x}_{(0,1,1)}$ comes before $\boldsymbol{x}_{(1,0,0)}$. Define the distance between two hexes X_1 and X_2 to be the L^2 distance between their vertex positions, i.e., $\|X_1 - X_2\|_F$.

If the hex specified by vertices X is invalid, we repair it by alternating between the following two steps until X is valid. First, we solve for the location $\boldsymbol{u}^*$ of the minimal Jacobian determinant via the moment relaxation (SOSD). Then, we solve the following nonlinear optimization to move the vertices X until $\boldsymbol{x}$ is locally injective at $\boldsymbol{u}^*$.

$$X^{k+1} = \left\{ \begin{array}{ll} \arg\min\limits_{X} & \|X - X^k\|_{\mathrm{F}}^2 \\ \text{s.t.} & \det(D\boldsymbol{x}_{\boldsymbol{u}^*}) \geq 0 \end{array} \right\} \tag{R}$$

We alternate these two steps until the SOS relaxation (SOSP) certifies that X is a valid hex element. In practice, we solve (R) using the default settings of Yalmip's nonlinear optimizer. See Table 1 for a summary of iterations taken to repair hex elements of different meshes. We summarize the single hex repair procedure in Algorithm 1.

5.2. Hex Mesh Repair

We now extend our simple repair algorithm to the entire hex mesh. Schematically, we wish to deform a given mesh into a valid mesh while also maintaining closeness to the original mesh, where closeness is measured by L^2 distance between corresponding vertex positions. To that end, let $\boldsymbol{V} \in \mathbb{R}^{n \times 3}$ be the matrix of vertex positions

Algorithm 1 Hex Element Repair $\Pi(X)$

procedure $\Pi(X_0)$
  $k \leftarrow 1$
  $X^k \leftarrow X_0$
  while X is not valid **do**
    $\boldsymbol{\mu}^* \leftarrow \arg\min_{\boldsymbol{\mu}}$ (SOSD) given X^k
    $\boldsymbol{u}^* \leftarrow (\mu_u^*, \mu_v^*, \mu_w^*)$
    $X^{k+1} \leftarrow$ SOLVE-(R)$(\boldsymbol{u}^*, X^k)$
    $k \leftarrow k+1$
  end while
  return X^{k+1}
end procedure

Algorithm 2 Hex Mesh Repair

procedure REPAIR-HEX-MESH(V_0, ρ)
  $k \leftarrow 1$
  $V^k \leftarrow V_0$
  while invalid hexes remain **do**
    $Z_\eta^k \leftarrow \Pi(H_\eta \boldsymbol{X}^k), \ \forall \eta \in \{1, \cdots, m\}$
    $V_i^{k+1} \leftarrow \frac{1}{1+\rho \cdot \deg_i} \left(V_i^0 + \rho \sum_{\eta \sim i} \boldsymbol{z}_{\eta i}^k \right)$
    $k \leftarrow k+1$
  end while
  return V^{k+1}
end procedure

for all vertices in the mesh. For each hexahedron $\eta \in \{1, \ldots, m\}$, let $H_\eta \in \{0,1\}^{8 \times n}$ select the vertices that appear in hex η in lexicographic order. So for each η, $H_\eta V \in \mathbb{R}^{8 \times 3}$ denotes the ordered set of vertices of hex η.

With a balancing parameter ρ, our overall hex mesh repair problem takes the form

$$\min_V \ \|V - V^0\|_{\mathrm{F}}^2 + \rho \sum_\eta \|\Pi(H_\eta V) - H_\eta V\|_F^2, \tag{21}$$

where $\Pi(\cdot)$ denotes the repair operation described in § 5.1.

To find a local minimum of (21), we alternate between repairing individual hexes $H_\eta V$ and aggregating them together into V. We detail the two alternating steps below and summarize our algorithm in Algorithm 2. We find that $\rho = 200$ is sufficient to achieve high-quality results in all our experiments.

Hex repair step. In this step, we split V into m independent 8×3 hex element matrices to be repaired in parallel, using the method in § 5.1. Let Z_η^k be the repair of element η, i.e., $Z_\eta^k = \Pi(H_\eta V^k)$.

Vertex update step. We now aggregate the individually-repaired hexes Z_η to update V. Using V_i to denote the position of vertex i, we compute

$$V_i^{k+1} := \frac{1}{1 + \rho \cdot \deg_i} \left(V_i^0 + \rho \sum_{\eta \sim i} \boldsymbol{z}_{\eta i}^k \right), \tag{22}$$

where $\deg_i$ is the number of hexes adjoining vertex i, $\eta \sim i$ denotes hexes η containing vertex i, and $\boldsymbol{z}_{\eta i}^k$ is the row of Z_η^k corresponding to vertex i. The updated vertex gets set to a weighted average of its initial position and its positions in the repaired hexes.

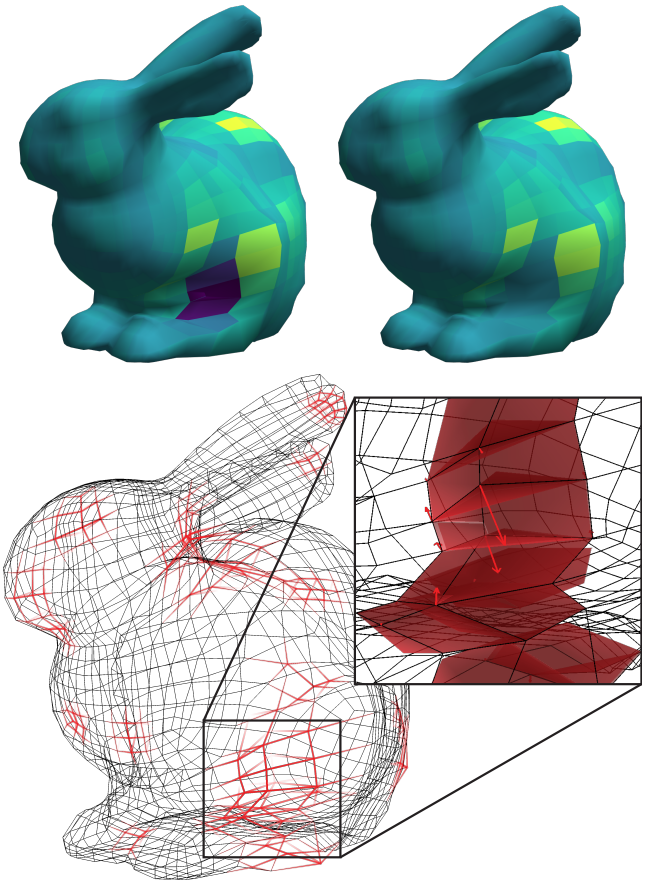

Figure 7: *The **bunny** example mesh from [Tak19], accessed via [BPLC19]. (Top left) The original mesh, with each hex colored based on its minimum Jacobian determinant, where purple indicates a low value and yellow indicates a high value. (Top right) The resultant mesh after running our repair algorithm, on the same color map. Note that the previously invalid elements are now valid. The empirical results of this experiment are show in Table 1. (Bottom) We highlight points that moved more throughout the repair algorithm in brighter red. The inset shows the most invalid section of the hex, with invalid elements highlighted in red and vectors representing the direction each vertex moved from the original mesh to the repaired mesh.*

Figure 7 depicts the results of our algorithm on the **bunny** mesh from [Tak19], which initially has inversions severe enough that boundary self-intersections are visible.

6. Experiments

In the previous section, we leveraged our ability to pinpoint invalidities inside a hex element to derive an algorithm that repairs any such invalidities. In this section, we demonstrate the capabilities of our invalidity detection and hex repair algorithms empirically.

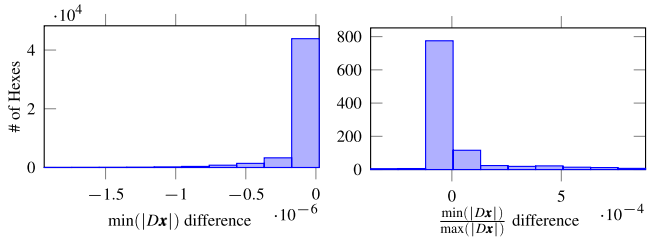

Figure 8: *(Left) Histogram of the difference between the minimum Jacobian determinant calculated by the SOS relaxation and the true minimum value determined by dense sampling on 50000 randomly generated cubes. (Right) Histogram comparing the ratio of minimum deteriminant to maximum determinant calculated by our algorithm to the ratio calculated by [JWR17] via [GR09]. This ratio is useful for applications that accept completely inverted elements.*

6.1. Hex Validity

Exact recovery. We begin with empirical validation that our SOS degree is high enough for exact recovery by testing our algorithm on randomly sampled hexes. We sample a mix of valid and invalid hexes by perturbing the vertices of the unit cube by an isotropic Gaussian with $\sigma = 0.3$. Results of this experiment are visualized in Figure 8. First, we compare the minimum Jacobian determinant computed by our SOS relaxation with the minimum Jacobian determinant computed by densely sampling the hex. For a sufficiently dense sampling of the hex, we divide the parameter cube into 100^3 smaller cubes and find the sub-cube whose center has lowest Jacobian determinant. Then we further divide that sub-cube into another 100^3 cubes. We verify that the difference between our SOS computed minimum Jacobian and the densely sampled minimum Jacobian is less than 1×10^{-7} for 50000 hexes. We also compare results extracted by our method to those computed using [JWR17] as implemented in Gmsh [GR09] and show that the difference is below numerical precision on 1000 randomly sampled hexes. On these hexes the average time our method takes to compute the minimal Jacobian determinant and its location is 0.022 seconds. While our method is slower than runtimes reported by [JWR17], our method is additionally able to pinpoint the most invalid location within a hex.

Numerical validity. By applying our method to hexahedral meshes, we are able to detect invalid hex elements within them. This is visualized on the **torus** mesh from [CC19] in Figure 6. We find that many elements in this mesh are invalid: $\min_{\boldsymbol{u}} \det(D\boldsymbol{x}_{\boldsymbol{u}}) \leq 0$. Invalid hexes on the **torus** are concentrated near mesh singularities, which typically feature high distortion. In addition to detecting invalidities, we are also able to repair them via Algorithm 2.

Figure 9 depicts the results of our hex invalidity detection algorithm on the **bunny** and **torus** meshes in the form of histograms of minimal Jacobian determinant values per hex. The histograms indicate that both meshes include inverted hexes. In addition to detection of invalid hexes, this diagram serves as a diagnostic tool for general quality of a hex mesh. Using our algorithm, we can

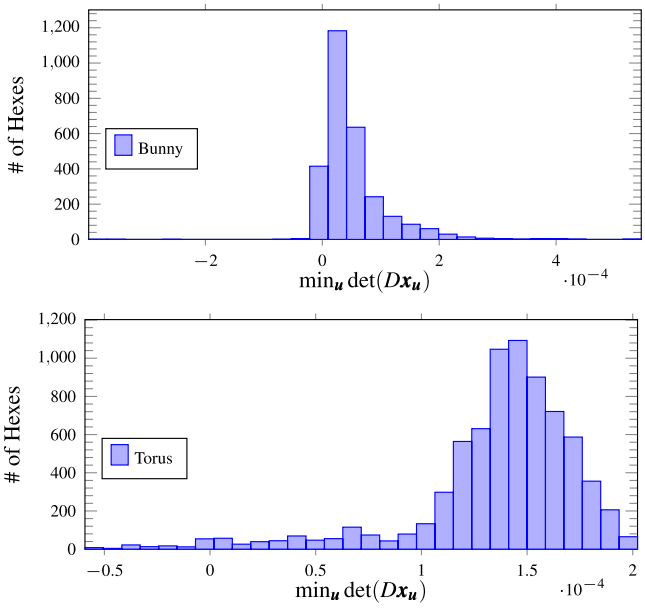

Figure 9: *Histograms of minimum Jacobians of hex elements for the **bunny** and **torus** meshes respectively. Note that both meshes exhibit inverted elements, which is detected by our algorithm. In addition to detection of invalid hexes, this diagram serves as a diagnostic tool for general quality of a hex mesh. Using our algorithm, we can build this aggregated view of hex element quality allowing users to ensure that the quality of their least valid hex is bounded.*

compute such an aggregated view of hex element quality allowing users to ensure the quality of their least valid hex is above a desired threshold.

Most distorted point. We use our SOS relaxation of the hex validity problem to compute the minimum Jacobian determinant on various hexahedra. A few illustrative examples are shown in Figure 1. For visualization purposes we sample the Jacobian determinant throughout the hex, but its minimal value is extracted by the SOS relaxation (SOSD). In addition to finding the minimal value, we show in Figure 4 that our method can extract the arg min via Equation (20). This is guaranteed theoretically by exact recovery, and visually by inspection of the colormap of the Jacobian determinant in Figure 4. Note that the minimal Jacobian determinant can occur on faces of the hex—thus any method that only checks corners is insufficient. We repeat this experiment on a larger scale using 50000 hexahedra and visualize their aggregated argmin locations on the parameter cube in Figure 10. This experiment reveals a fascinating pattern: that the minimal Jacobian determinant consistently lies on the boundary of the parameter cube. In addition to empiricially verifying Knupp's conjecture [Knu90], this suggests a hidden multi-linearity in the Jacobian determinant or a form of the maximum principle. We discuss this further in § 7 and leave its exploration to future work. On the other hand, this experiment reveals yet again that the most-distorted point may lie on a face, rather than on a corner. Therefore, any methods that only check validity on corners are insufficient for verifying hex validity.

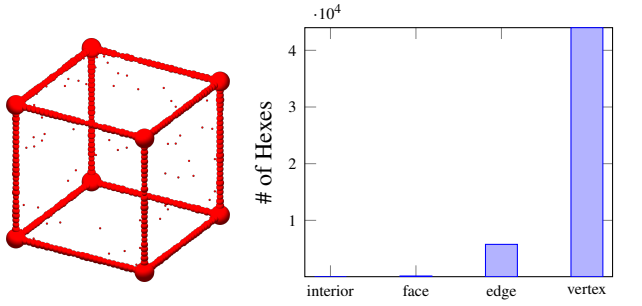

Figure 10: *We sample 50000 randomly generated valid and invalid hexes and compute the location of the minimal Jacobian determinant. (Left) We plot their locations $\boldsymbol{u}^*$ as a histogram on the parameter cube, where at the center of each bin we plot a sphere with the radius scaled on a log scale based on the number of points that fall into that bin. This shows a distinct pattern where the majority of $\boldsymbol{u}^*$ values are on edges of the parameter cube. (Right) We bucket the $\boldsymbol{u}^*$ values into four categories: interior, face, edge, and vertex. While the majority of $\boldsymbol{u}^*$ lie on corners, a significant amount lie on edges and faces. Within a tolerance of 1×10^{-4} we find no values of $\boldsymbol{u}^*$ that lie on the interior of the cube. This empirically verifies Knupp's conjecture [Knu90] (see § 1).*

6.2. Mesh Repair

In Figure 7 we can observe the method by which our algorithm repairs a mesh, by applying small perturbations to the vertices in order to find valid hexes with minimal vertex movement. While the original mesh does not represent a physically realizable domain, the repaired mesh does. Termination of our hex mesh repair algorithm certifies local injectivity.

We show the progress of our hex mesh repair algorithm in Figure 11 on the **torus** and **bunny** meshes. The blue curve associated to the left y-axis indicates that the number of invalid hexes decreases monotonically. The red curve associated to the right y-axis indicates that the minimum Jacobian determinant increases monotonically. Both meshes are repaired within 20 iterations of Algorithm 2. Our algorithm detects no further invalidities, certifying injectivity of the repaired meshes.

In Figure 12 we show the successful repair of a variety of meshes from [BPLC19]. In the first column, invalid hexes of the input meshes are highlighted in red. In the middle column, edges that move over the course of repair are highlighted to indicate their displacement. The final column shows the minimum Jacobian determinant after repair. Our algorithm succeeds in repairing all input meshes by concentrating perturbations in the neighborhoods of invalid hexes. Input hexes that were valid are left relatively untouched, and the repaired meshes are valid.

Figure 14 depicts three meshes—the **fertility** from [XLZ*19] and the **cat** and **dolphin** from [GMD*16]—featuring invalid elements that cannot be detected by checking Jacobians at corners alone. The arrows (not to scale) show how our mesh repair Algorithm 2 displaces vertices to fully repair the meshes.

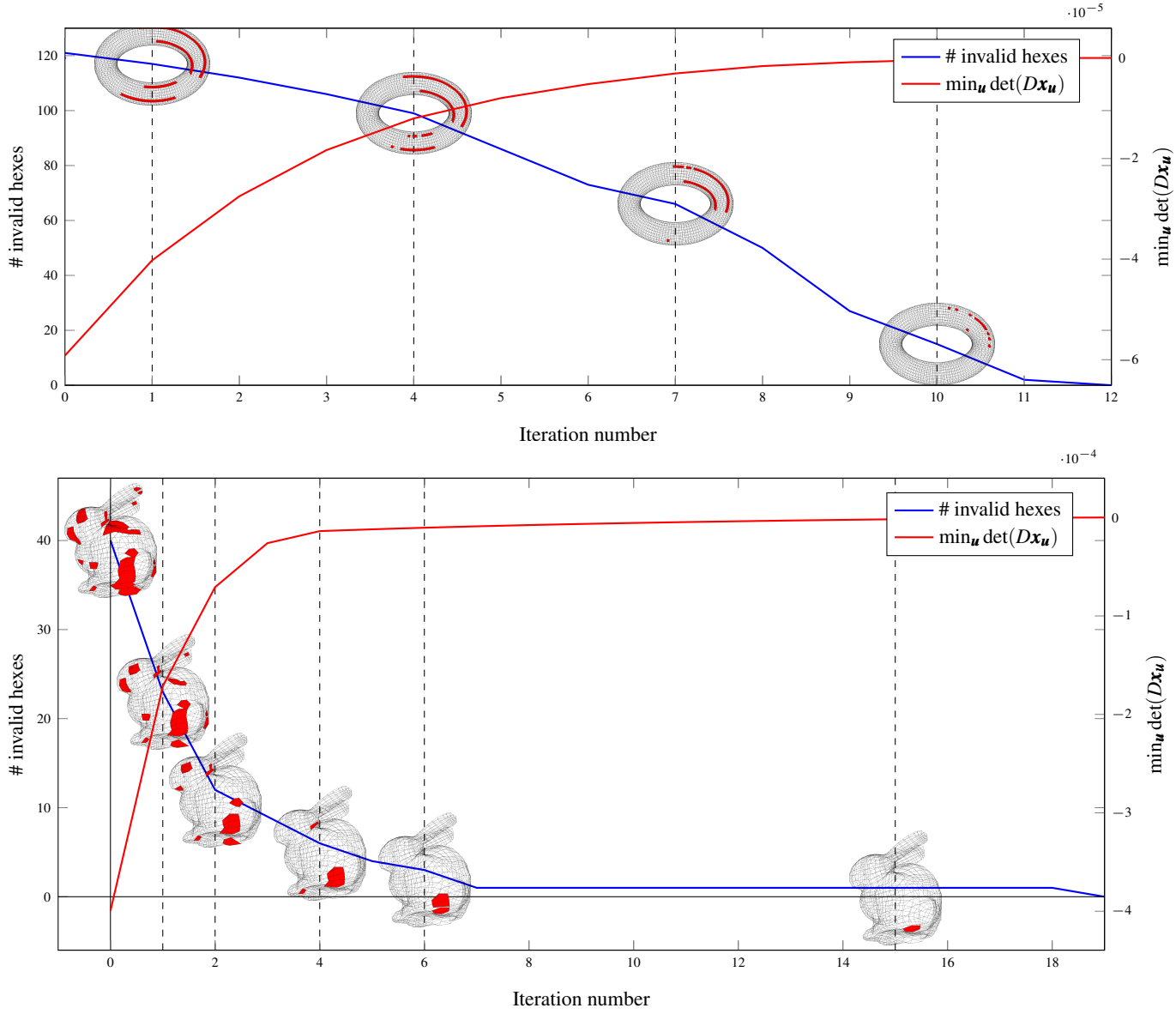

Figure 11: *Our mesh repair Algorithm 2 corrects the* ***torus*** *mesh from [CC19] in only* 12 *iterations and the* ***bunny*** *mesh from [Tak19] in only 19 iterations. The number of invalid hexes remaining in the mesh is plotted in blue and is associated to the left y-axis. We also plot the minimum Jacobian determinant in red and associate it to the right y-axis. In both cases the number of invalid hexes decreases monotonically, and the minimum Jacobian determinant increases monotonically.*

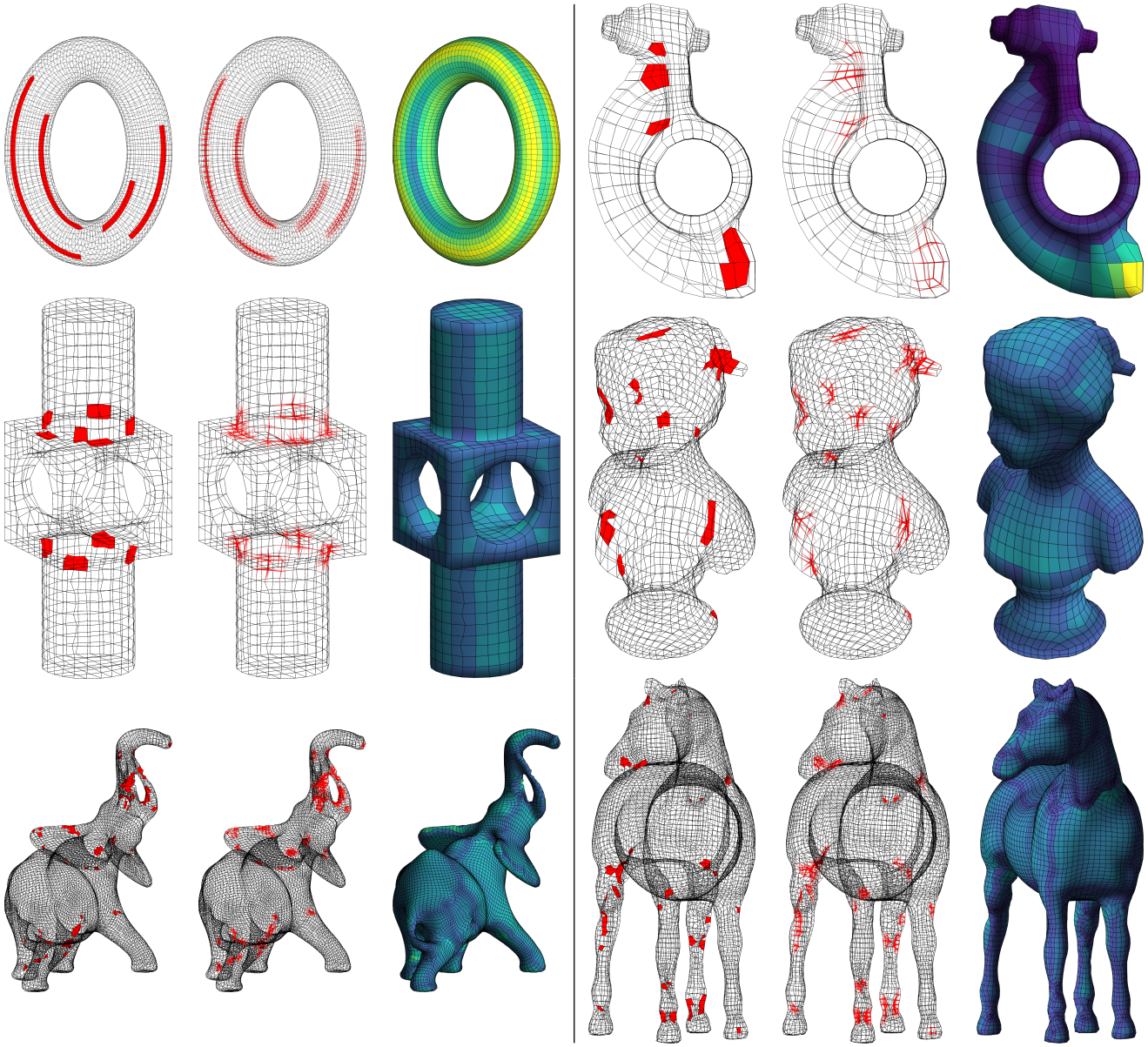

Figure 12: *(Left column) Input hex mesh with invalid hexes highlighted in red. (Middle column) Edges are highlighted in red by how much they move during our mesh repair procedure. This visualization shows that the majority of movement is concentrated on edges adjacent to invalid hexes, and that very little change is made to valid regions, as one might hope. (Right column) We show the Jacobian determinant as a colormap over the fixed hex mesh. Meshes included here are* ***torus*** *and* ***rockerarm*** *from [BPLC19] and* ***horse***, ***block***, ***elephant***, *and* ***bust*** *from [XGC17]. We use* $\rho = 200$ *for all tests.*

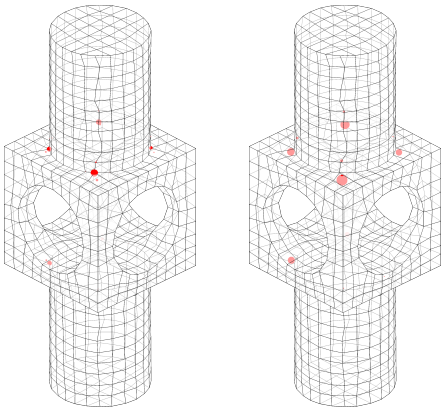

Figure 13: *When repaired using the procedure detailed in § 5.2, the boundary vertices can move (left). The procedure can be adjusted to pin the boundary vertices in place (right). The red spheres are centered at the final positions of vertices that move, and their radii indicate the total displacement of those vertices.*

Modifying the nonlinear hex element repair step $\Pi(X)$ so that boundary vertices are fixed yields repaired meshes that preserve boundary geometry when such a solution exists (see Figure 13).

6.3. Comparison to Previous Methods

In Table 1 we show aggregate statistics of our hex mesh repair algorithm. We show the number of inner iterations (iterations of Algorithm 1), number of outer iterations (iterations of Algorithm 2), number of SOS optimizations solved, and total repair time. In addition we compare the success of our mesh repair algorithm to the method in [XGC17], which also aims to untangle hex meshes and use non-inverting deformations to ensure injectivity. We find that their method can exhibit instability on certain meshes. In contrast, our mesh repair algorithm succeeds on all test meshes, and guarantees injectivity through the theory of SOS relaxations.

7. Discussion and Conclusion

This work represents an important step toward broader application of SOS relaxations in geometry processing, where both polynomials and optimization are ubiquitous. Through this general machinery, we enable users of hex meshes in 3D modeling and simulation not only to quantify the validity of hex elements, but also to repair invalid hex meshes. Our experiments reveal that many hex meshes produced by modern automatic hex meshing algorithms have invalid elements that would disqualify them from use in finite element analysis. While these problems might have previously gone unnoticed, our mesh repair method can breathe new life into these hex meshes.

There are a number of immediate extensions that take advantage of the generality of SOS relaxation. For example, the Jacobian determinant is not only a polynomial in the parameters (u,v,w), but simultaneously in the vertex positions $\boldsymbol{x}_{ijk}$. Suppose one could compute the smallest value Λ that $\det(D\boldsymbol{x})$ can take at a vertex such that

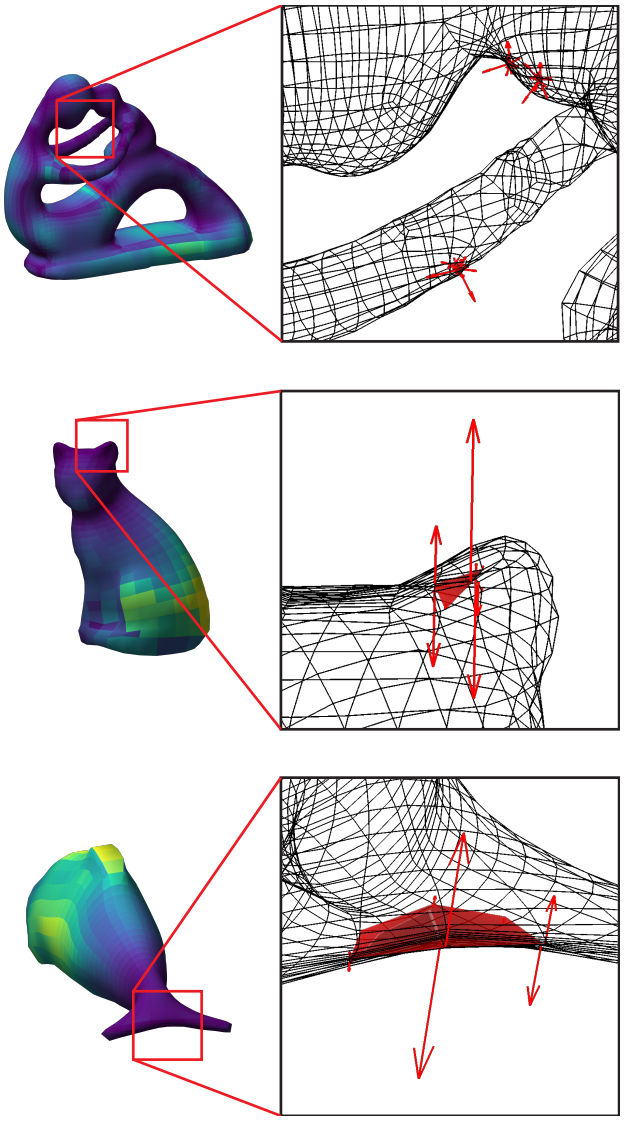

Figure 14: *We show the success of our repair algorithm run on the **fertility**, **cat**, and **dolphin** meshes from [BPLC19], all three of which have no hexes that are invalid at the corners. The highlighted red elements have invalidities in their interior, which we are able to detect with our SOS relaxation. We also show overlaid on the original mesh the direction in which our repair algorithm moves each vertex.*

Mesh	# hexes	# invalid	# outer iters	# (SOSD) steps	# (R) steps	Repair time (min)	# invalid after [XGC17]
bunny	2832	40	19	8173	123	4.05	0
torus	7380	121	12	21,157	1471	12.13	N/A
bust	5258	30	49	20,020	643	9.20	0
block	2520	19	11	5353	83	2.29	0
rockerarm	1858	11	7	2616	47	1.24	0
cat	2604	1	12	2818	22	1.41	4
fertility	21,016	20	21	24,998	1168	9.83	0
dolphin	4788	2	2	4838	3	2.45	8
horse	44,145	73	12	53,986	457	22.15	0
elephant	46,525	90	11	60,159	301	24.85	0

Table 1: *We compare our hex mesh repair method to [XGC17], which untangles meshes with boundary relaxations and non-inverting deformations. For our repair algorithm, we show aggregate statistics about our optimization, and runtime in seconds. For their algorithm, we show if their algorithm successfully terminated, as well as the number of invalid hexes remaining at the end of their algorithm. While their method sporadically fails to terminate or fails to repair the hex mesh, our algorithm succeeds on all test cases.*

the hex is still valid. Then checking whether $\det(D\boldsymbol{x}) \geq \Lambda$ at vertices would provide a conservative validity guarantee. This would save a great deal of computation as SDPs would only need to be solved for the few elements that are likely to be invalid. The problem of finding Λ can be formulated as a polynomial optimization problem and approached with the methods of SOS relaxation.

Another extension of our methodology would be to detect and repair invalidities in different element types, including those of higher polynomial degree. In theory this only results in a different degree for the determinant, and/or support on a different reference element. High level polynomial optimization tools like [Löf04] support modeling such extensions.

Similarly, the projection Π that we approximate (see § 5) can be formulated as a polynomial optimization problem and relaxed using SOS. Assuming it admits an exact relaxation of low enough degree, we could use this convex projection to yield global guarantees for the alternating mesh repair optimization in § 5 or a similar ADMM method.

Knupp [Knu90] has conjectured that it is sufficient to check only the faces of a hex for validity. Our empirical results strongly confirm this conjecture—the minimum Jacobian determinant occurs on the boundary of every hex we have examined. In view of this evidence, we expect it should not be too difficult to prove Knupp's conjecture formally. Verifying this conjecture would speed up hex validity checks by reducing the degrees of polynomials involved.

By placing the hex validity problem within the framework of SOS programming, we introduce the full power of this general machinery to computer graphics applications. We hope that the geometry processing community will find more exciting uses for it wherever polynomials and optimization appear together.

Acknowledgements

The authors would like to thank Pablo Parrilo for his clear notes on SOS programming [Par19], as well as Kaoji Xu [XGC17] and Amaury Johnen [JWR17] for their open source code implementations.

David Palmer appreciates the generous support of the Fannie and John Hertz Foundation through the Hertz Graduate Fellowship. Paul Zhang acknowledges the generous support of the Department of Energy Computer Science Graduate Fellowship and the MathWorks Fellowship. Justin Solomon and the MIT Geometric Data Processing group acknowledge the generous support of Army Research Office grants W911NF1710068 and W911NF2010168, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grant IIS-1838071, of Navy STTR 2020.A (topic N20A-T004), from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, and from the Skoltech–MIT Next Generation Program.

References

[AHMS17] AHMADI A., HALL G., MAKADIA A., SINDHWANI V.: Geometry of 3d environments and sum of squares polynomials. doi:10.15607/RSS.2017.XIII.071. 3

[Ali95] ALIZADEH F.: Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization 5*, 1 (1995), 13–51. 3

[ApS19] APS M.: *The MOSEK Fusion API for C++ manual. Version 9.0.*, 2019. URL: https://docs.mosek.com/9.0/cxxfusion/index.html. 6

[BBV04] BOYD S., BOYD S. P., VANDENBERGHE L.: *Convex Optimization*. Cambridge University Press, 2004. 3

[BCY11] BRANDÃO F. G., CHRISTANDL M., YARD J.: A quasipolynomial-time algorithm for the quantum separability problem. In *ACM Symposium on Theory of Computing* (2011), pp. 343–352. 3

[BDK*03] BREWER M., DIACHIN L., KNUPP P., LEURENT T., MELANDER D.: The Mesquite mesh quality improvement toolkit. 2

[BM03] BURER S., MONTEIRO R. D.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming 95*, 2 (2003), 329–357. 3

[BPLC19] BRACCI M., PIETRONI N., LIVESU M., CIGNONI P.: Hexalab.net: An online viewer for hexahedral meshes. *Computer-Aided Design 110* (05 2019). doi:10.1016/j.cad.2018.12.003. 2, 7, 8, 9, 11, 12

[BPM*95] BENZLEY S., PERRY E., MERKLEY K., CLARK B.,

SJAARDEMA G.: A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. *International Meshing Roundtable 17* (01 1995). 1

[BPT12] BLEKHERMAN G., PARRILO P. A., THOMAS R. R.: *Semidefinite Optimization and Convex Algebraic Geometry*. SIAM, 2012. 3, 4

[BTEGN09] BEN-TAL A., EL GHAOUI L., NEMIROVSKI A.: *Robust Optimization*, vol. 28. Princeton University Press, 2009. 3

[BVB18] BOUMAL N., VORONINSKI V., BANDEIRA A. S.: Deterministic guarantees for Burer-Monteiro factorizations of smooth semidefinite programs. *Communications on Pure and Applied Mathematics* (2018). 3

[CC19] CORMAN E., CRANE K.: Symmetric moving frames. *ACM Trans. Graph. 38*, 4 (July 2019). URL: https://doi.org/10.1145/3306346.3323029, doi:10.1145/3306346.3323029. 1, 7, 8, 10

[CK92] CIFUENTES A., KALBAG A.: A performance study of tetrahedral and hexahedral elements in 3-d finite element structural analysis. *Finite Elements in Analysis and Design 12*, 3-4 (1992), 313–318. 1

[CM19] CIFUENTES D., MOITRA A.: Polynomial time guarantees for the Burer-Monteiro method, 2019. arXiv:1912.01745. 3

[DOS99] DEY S., O'BARA R. M., SHEPHARD M. S.: Curvilinear mesh generation in 3d. In *IMR* (1999), Citeseer, pp. 407–417. 2

[FM99] FOLWELL N., MITCHELL S.: Reliable whisker weaving via curve contraction. *Engineering With Computers 15* (01 1999), 292–302. doi:10.1007/s003660050024. 1

[FXBH16] FANG X., XU W., BAO H., HUANG J.: All-hex meshing using closed-form induced polycube. *ACM Transactions on Graphics 35*, 4 (July 2016), 1–9. URL: http://dl.acm.org/citation.cfm?doid=2897824.2925957, doi:10.1145/2897824.2925957. 1

[GHX*17] GAO X., HUANG J., XU K., PAN Z., DENG Z., CHEN G.: Evaluating hex-mesh quality metrics via correlation analysis. *Computer Graphics Forum 36* (08 2017), 105–116. doi:10.1111/cgf.13249. 2

[GMD*16] GAO X., MARTIN T., DENG S., COHEN E., DENG Z., CHEN G.: Structured volume decomposition via generalized sweeping. *IEEE Transactions on Visualization and Computer Graphics 22*, 7 (2016), 1899–1911. 9

[GPW*17] GAO X., PANOZZO D., WANG W., DENG Z., CHEN G.: Robust structure simplification for hex re-meshing. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–13. 2

[GR09] GEUZAINE C., REMACLE J.-F.: Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering 79*, 11 (2009), 1309–1331. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579, doi:10.1002/nme.2579. 8

[Gra99] GRANDY J.: Conservative remapping and region overlays by intersecting arbitrary polyhedra. *Journal of Computational Physics 148*, 2 (1999), 433 – 466. URL: http://www.sciencedirect.com/science/article/pii/S0021999198961253, doi:https://doi.org/10.1006/jcph.1998.6125. 2

[GV01] GRIGORIEV D., VOROBJOV N.: Complexity of null-and positivstellensatz proofs. *Annals of Pure and Applied Logic 113*, 1-3 (2001), 153–160. 3, 4

[GW95] GOEMANS M. X., WILLIAMSON D. P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM) 42*, 6 (1995), 1115–1145. 3

[HG13] HUANG Q.-X., GUIBAS L.: Consistent shape maps via semidefinite programming. In *Proc. Symposium on Geometry Processing* (2013), Eurographics Association, pp. 177–186. 3

[HJS*14] HUANG J., JIANG T., SHI Z., TONG Y., BAO H., DESBRUN M.: ℓ_1-based construction of polycube maps from complex shapes. *ACM Transactions on Graphics 33*, 3 (June 2014), 1–11. URL: http://dl.acm.org/citation.cfm?doid=2631978.2602141, doi:10.1145/2602141. 1

[HMESM06] HERNANDEZ-MEDEROS V., ESTRADA-SARLABOUS J., MADRIGAL D. L.: On local injectivity of 2d triangular cubic Bézier functions. *Investigación Operacional 27*, 3 (2006), 261–276. 2

[HTWB11] HUANG J., TONG Y., WEI H., BAO H.: Boundary aligned smooth 3d cross-frame field. In *SIGGRAPH Asia* (Hong Kong, China, 2011), ACM Press, p. 1. URL: http://dl.acm.org/citation.cfm?doid=2024156.2024177, doi:10.1145/2024156.2024177. 1

[HZ16] HU K., ZHANG Y. J.: Centroidal Voronoi tessellation based polycube construction for adaptive all-hexahedral mesh generation. *Computer Methods in Applied Mechanics and Engineering 305* (2016), 405–421. 1

[HZX18] HU K., ZHANG Y. J., XU G.: CVT-based 3d image segmentation and quality improvement of tetrahedral/hexahedral meshes using anisotropic Giaquinta-Hildebrandt operator. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 6*, 3 (2018), 331–342. 2

[JRG12] JOHNEN A., REMACLE J.-F., GEUZAINE C.: Geometrical validity of curvilinear finite elements. In *International Meshing Roundtable* (Berlin, Heidelberg, 2012), Quadros W. R., (Ed.), Springer Berlin Heidelberg, pp. 255–271. 2

[JWR17] JOHNEN A., WEILL J.-C., REMACLE J.-F.: Robust and efficient validation of the linear hexahedral element. *Procedia Engineering 203* (2017), 271–283. URL: http://dx.doi.org/10.1016/j.proeng.2017.09.809, doi:10.1016/j.proeng.2017.09.809. 2, 8, 13

[Knu90] KNUPP P. M.: On the invertibility of the isoparametric map. *Computer Methods in Applied Mechanics and Engineering 78*, 3 (1990), 313 – 329. URL: http://www.sciencedirect.com/science/article/pii/0045782590900046, doi:https://doi.org/10.1016/0045-7825(90)90004-6. 2, 9, 13

[Knu99] KNUPP P. M.: Winslow smoothing on two-dimensional unstructured meshes. *Engineering with Computers 15*, 3 (1999), 263–268. 2

[Knu00] KNUPP P. M.: Hexahedral mesh untangling and algebraic mesh quality metrics. In *IMR* (2000), Citeseer, pp. 173–183. 2

[Knu03] KNUPP P. M.: Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elem. Anal. Des. 39*, 3 (Jan. 2003), 217–241. URL: https://doi.org/10.1016/S0168-874X(02)00070-7, doi:10.1016/S0168-874X(02)00070-7. 2

[Las01] LASSERRE J. B.: Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization 11*, 3 (2001), 796–817. 3, 4, 6

[Ler08] LEROY R.: *Certificates of positivity and polynomial minimization in the multivariate Bernstein basis.* Theses, Université Rennes 1, Dec. 2008. URL: https://tel.archives-ouvertes.fr/tel-00349444. 2

[Ler09] LEROY R.: Certificates of positivity in the simplicial Bernstein basis. working paper or preprint, 2009. URL: https://hal.archives-ouvertes.fr/hal-00589945. 2

[Löf04] LÖFBERG J.: YALMIP: A toolbox for modeling and optimization in MATLAB. In *In Proceedings of the CACSD Conference* (Taipei, Taiwan, 2004). 6, 13

[LSVT15] LIVESU M., SHEFFER A., VINING N., TARINI M.: Practical hex-mesh optimization via edge-cone rectification. vol. 34. doi:10.1145/2766905. 2

[LXZQ13] LENG J., XU G., ZHANG Y., QIAN J.: *Quality Improvement of Segmented Hexahedral Meshes Using Geometric Flows*, vol. 3. 01 2013, pp. 195–221. doi:10.1007/978-94-007-4255-0_11. 2

[LZC*18] LIU H., ZHANG P., CHIEN E., SOLOMON J., BOMMES D.: Singularity-constrained octahedral fields for hexahedral meshing. *ACM Transactions on Graphics 37*, 4 (July 2018), 1–17. URL: http://dl.acm.org/citation.cfm?doid=3197517.3201344, doi:10.1145/3197517.3201344. 1

[MDK*16] MARON H., DYM N., KEZURER I., KOVALSKY S., LIPMAN Y.: Point registration via efficient convex relaxation. *ACM Trans. Graph. 35*, 4 (2016), 73. 3

[NN94] NESTEROV Y., NEMIROVSKII A.: *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611970791, arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611970791, doi:10.1137/1.9781611970791. 3

[NRP11] NIESER M., REITEBUCH U., POLTHIER K.: CubeCover–parameterization of 3d volumes. *Computer Graphics Forum 30*, 5 (Aug. 2011), 1397–1406. URL: http://doi.wiley.com/10.1111/j.1467-8659.2011.02014.x, doi:10.1111/j.1467-8659.2011.02014.x. 1

[Par00] PARRILO P. A.: *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000. 3

[Par19] PARILLO P.: Algebraic techniques and semidefinite optimization. https://learning-modules.mit.edu/materials/index.html?uuid=/course/6/sp19/6.256#materials, 2019. 13

[PBS20] PALMER D., BOMMES D., SOLOMON J.: Algebraic representations for volumetric frame fields. *ACM Trans. Graph. 39*, 2 (Apr. 2020). URL: https://doi.org/10.1145/3366786, doi:10.1145/3366786. 1, 3

[PP02] PAPACHRISTODOULOU A., PRAJNA S.: On the construction of Lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* (2002), vol. 3, IEEE, pp. 3482–3487. 3

[Put93] PUTINAR M.: Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal 42*, 3 (1993), 969–984. 3, 4

[QZ12] QIAN J., ZHANG Y.: Automatic unstructured all-hexahedral mesh generation from B-reps for non-manifold CAD assemblies. *Engineering with Computers 28*, 4 (2012), 345–359. 1

[QZW*10] QIAN J., ZHANG Y., WANG W., LEWIS A. C., QIDWAI M. S., GELTMACHER A. B.: Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. *International journal for numerical methods in engineering 82*, 11 (2010), 1406–1423. 2

[RGRS14] RUIZ-GIRONÉS E., ROCA X., SARRATE J.: Optimizing mesh distortion by hierarchical iteration relocation of the nodes on the cad entities. *Procedia Engineering 82* (12 2014). doi:10.1016/j.proeng.2014.10.376. 2

[SEK*07] STIMPSON C., ERNST C., KNUPP P., PÉBAY P., THOMPSON D.: The verdict geometric quality library. *Sandia National Laboratories, SAND2007-1751* (2007). 2

[Sin11] SINGER A.: Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis 30*, 1 (2011), 20. 3

[SVB17] SOLOMON J., VAXMAN A., BOMMES D.: Boundary element octahedral fields in volumes. *ACM Transactions on Graphics 36*, 3 (May 2017), 1–16. URL: http://dl.acm.org/citation.cfm?doid=3087678.3065254, doi:10.1145/3065254. 1

[Tak19] TAKAYAMA K.: Dual sheet meshing: An interactive approach to robust hexahedralization. *Comput. Graph. Forum 38* (2019), 37–48. 8, 10

[TGRL13] TOULORGE T., GEUZAINE C., REMACLE J.-F., LAMBRECHTS J.: Robust untangling of curvilinear meshes. *Journal of Computational Physics 254* (2013), 8–26. 2

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: PolyCube-Maps. In *Proc. SIGGRAPH* (Los Angeles, California, 2004), ACM Press, p. 853. URL: http://portal.acm.org/citation.cfm?doid=1186562.1015810, doi:10.1145/1186562.1015810. 1

[Ush01] USHAKOVA O.: Conditions of nondegeneracy of three-dimensional cells. a formula of a volume of cells. *SIAM Journal on Scientific Computing 23* (01 2001). doi:10.1137/S1064827500380702. 2

[Ush11] USHAKOVA O. V.: Nondegeneracy tests for hexahedral cells. *Computer Methods in Applied Mechanics and Engineering 200*, 17 (2011), 1649 – 1658. URL: http://www.sciencedirect.com/science/article/pii/S0045782511000247, doi:https://doi.org/10.1016/j.cma.2011.01.014. 2

[Vav03] VAVASIS S.: A Bernstein-Bézier sufficient condition for invertibility of polynomial mapping functions. 2

[Wei94] WEINGARTEN V. I.: The controversy over hex or tet meshing. *Machine design 66*, 8 (1994), 74–76. 1

[WSR*12] WILSON T., SARRATE J., ROCA X., MONTENEGRO R., ESCOBAR J.: Untangling and smoothing of quadrilateral and hexahedral meshes. vol. 100. doi:10.4203/ccp.100.36. 2

[XGC17] XU K., GAO X., CHEN G.: Hexahedral mesh quality improvement via edge-angle optimization. *Computers and Graphics 70* (07 2017). doi:10.1016/j.cag.2017.07.002. 2, 11, 12, 13

[XLZ*19] XU G., LING R., ZHANG Y., XIAO Z., JI Z., RABCZUK T.: Singularity structure simplification of hexahedral mesh via weighted ranking. *ArXiv abs/1901.00238* (2019). 9

[ZBX09] ZHANG Y., BAJAJ C., XU G.: Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. *Communications in Numerical Methods in Engineering 25*, 1 (2009), 1–18. 2

[ZDG96] ZHOU K., DOYLE J. C., GLOVER K.: *Robust and Optimal Control*, vol. 40. Prentice Hall New Jersey, 1996. 3

[Zha05] ZHANG S.: Subtetrahedral test for the positive Jacobian of hexahedral elements. 2