Sum-of-Squares Collision Detection for Curved Shapes and Paths

Paul Zhang*
pzpzpzp1@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA
Walt Disney Animation Studios
Burbank, CA, USA

Zoë Marschner*
zoem@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA
Walt Disney Animation Studios
Burbank, CA, USA

Justin Solomon
jsolomon@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

Rasmus Tamstorf
rasmus.tamstorf@disneyanimation.com
Walt Disney Animation Studios
Burbank, CA, USA

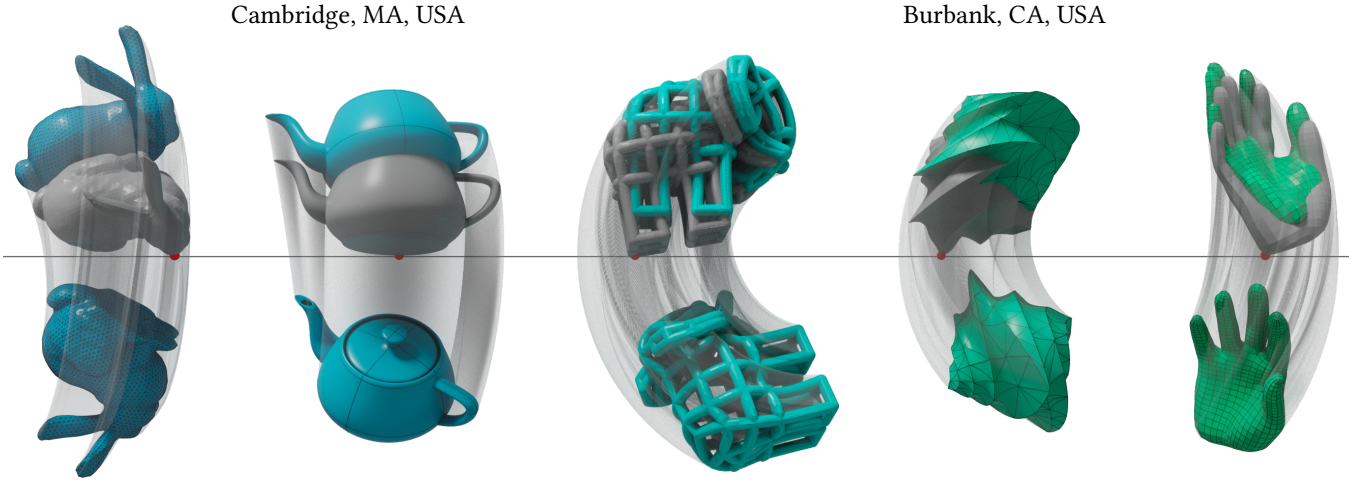


Figure 1: From left to right, we show a linear triangle mesh, bicubic quad mesh, cubic spline wireframe, quartic triangle mesh, and a trilinear hexahedral (hex) mesh. We solve continuous collision detection between these geometries moving along the rigid $\text{ScLERP}_{\text{Tay}}$ trajectory described in §4.4 and a plane. Our formulation with SOSP certifiably solves this hard problem and naturally generalizes to any polynomial geometric element type.

ABSTRACT

Sum-of-Squares Programming (SOSP) has recently been introduced to graphics as a unified way to address a large set of difficult problems involving higher order primitives. Unfortunately, a challenging aspect of this approach is the computational cost—especially for problems involving multiple geometries like collision detection. In this paper, we present techniques to reduce the cost of SOSP significantly. We use these improvements to speed up difficult problems like collision detection between Bézier triangles by as much as 300×. In addition, motivated by hair bundle simulation, we present SOSP based collision detection on the tapered cubic cylinder. We also present an algebraic formulation of rigid body motion enabling SOSP based collision detection for curved geometries and trajectories simultaneously. While these new formulations are complex, our speedups make them feasible. These advances improve the applicability of SOSP based collision detection and enable the continued progress of higher-order geometry processing.

CCS CONCEPTS

• **Computing methodologies** → *Parametric curve and surface models*; **Collision detection**; *Shape analysis.*

KEYWORDS

collision detection, sum-of-squares programming, parametric shape analysis

ACM Reference Format:
Paul Zhang, Zoë Marschner, Justin Solomon, and Rasmus Tamstorf. 2023. Sum-of-Squares Collision Detection for Curved Shapes and Paths. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 06–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3588432.3591507

*Both authors contributed equally to this research.

SIGGRAPH '23 Conference Proceedings, August 06–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08.
https://doi.org/10.1145/3588432.3591507

1 INTRODUCTION

Modern computer-aided design (CAD) and simulation tools support curved geometric modeling through a variety of parametric methods such as high-order polynomial or rational basis functions. Techniques like finite-element analysis (FEA) or isogeometric analysis (IGA) allow numerical analysis of curved geometries using high-order basis functions that respect the original curved geometry. The benefits of performing these analyses with high-order basis functions are well documented [Schneider et al. 2022]. High-order geometry has graphical benefits as well; as an example, production hair curves are often rendered as cubic splines for resolution independent smoothness [Burley et al. 2018; Kulla et al. 2018; Nakamaru and Ohno 2002]. Yet, in the context of dynamic simulation, piecewise linear geometry remains the most common [Bergou et al. 2010; Thyng et al. 2017].

In simulation, linearization of curved geometry introduces discrepancies that propagate into the simulation, decreasing its overall efficiency. Despite the drawbacks, linearization has a primary advantage explaining its pervasiveness: *collision detection for flat geometry is far easier than collision detection for curved geometry*. Collision detection for curved geometries frequently ends up as a generic nonlinear optimization problem with few guarantees, as opposed to collision detection between flat geometries, which is simpler and well studied with impressively optimized runtimes and guarantees [Provot 1997; Wang et al. 2021]. For this reason, dynamic simulation usually turns to linearization.

Marschner et al. [2021] tackle the problem of high-order geometry processing, presenting a Sum-of-Squares Programming (SOSP) based framework to solve several challenging problems on curved geometries. Among these is continuous collision detection (CCD), in which one seeks to find the earliest time of collision along a given trajectory. The existing SOSP based CCD solution, while effective, comes at a prohibitive runtime cost of up to 700 seconds for cubic Bézier triangles and is not able to self-certify correctness.

In this work, we propose techniques to accelerate the SOSP methodology and apply them to collision detection. Speedups are primarily obtained by pruning redundant monomials in the SOSP problem and re-expressing equivalent varieties in forms more amicable to optimization. Furthermore, we augment the SOSP formulation of collision detection by providing certificates for earliest-collision, intersecting-pair, and non-collision, which enables a number of culling steps. Using our optimized formulation, we are able to consider more complicated queries including collision between tapered cubic cylinders useful for hair rendering and collision between curved geometries following rigid body motions. To apply SOSP to the rigid body motion problem, we develop an algebraic formulation of rigid body motion based on dual quaternions.

2 RELATED WORK

Sum-of-Squares Programming. Blekherman et al. [2012] provide a comprehensive review of this field. Challenging polynomial positivity constraints can be convexified into Sum-of-Squares (SOS) constraints encoded by semidefinite matrices, ultimately producing a semidefinite program (SDP). SDPs are solvable in polynomial time via interior-point methods [Alizadeh 1995; Nesterov and Nemirovskii 1994]. In practice, modeling tools such as YALMIP [Löfberg 2004] convert SOSPs into SDPs that can then be solved by optimization software like MOSEK [MOSEK ApS 2020].

SOSP in Geometry Processing. Closest to our work is [Marschner et al. 2020, 2021], where SOSP is applied to a variety of geometric problems including continuous collision detection (CCD) between pairs of polynomial patches. Amice et al. [2023] use SOSP to map out non-colliding configurations for robotic arms. Yang et al. [2022] alternate convex optimization with rank-one matrix projections to improve runtime of SOSPs for point cloud correspondence. Ahmadi et al. [2017] use SOSP to encapsulate point clouds with level set surfaces.

High-Order Dynamics. Along with IGA [Hughes et al. 2005], high-order simulation has recently increased in popularity, including for contact mechanics [De Lorenzis et al. 2014]. In many cases, however, collisions are either ignored or handled via approximations like linearization that lack guarantees. Lu and Zheng [2014] consider NURBS based cloth simulation but use tessellation and a standard Newton's method for collision detection. Similarly, Trusty et al. [2021] use NURBS patches for volumetric elastic simulation but perform collision detection on a coarse linearization, thus failing to detect interpenetration of the actual curved geometries. Bertails et al. [2006] use piecewise helical elements to simulate hair dynamics. Ferguson et al. [2021] handle rigid body simulation with curved trajectories, but require flat geometry. Snyder et al. [1993] use Newton's method with interval arithmetic for curved geometry collision detection but need to be provided with convergent inclusion functions per geometry. Their method also specifically finds collision points and does not provide separation distance bounds that can be helpful for simulation. Finally Ferguson et al. [2022] propose a hybrid high-order simulation coupled to linear mesh collision handling, bypassing the need for exact curved geometry collision detection.

Algebraic Rigid Body Motions. Rigid body motions are curves in SE(3). An example of such a curve is a *screw motion*. Screw motions are characterized by jointly constant rotational and translational velocity. Unfortunately, screw motions are non-algebraic [Wampler and Sommese 2011], while SOSP is fundamentally limited to algebraic expressions. Redon et al. [2000] use nonlinear parameterizations of translation to produce an algebraic non-screw rigid body trajectory. Kavan et al. [2008] use dual quaternion linear interpolation to approximate pure screw motions. Paul [1997] uses dual quaternions for contact problems on flat polyhedra.

3 PRELIMINARIES

We summarize the most relevant background material below.

3.1 Sum-of-Squares Programming

Sum-of-squares programming is a technique for converting polynomial optimization problems into convex semidefinite programs. Here, we summarize the details of SOSP necessary to apply it to our collision detection problems. For a general summary of SOSP, see [Parrilo 2019]; for further exposition in the context of geometry processing, see [Marschner et al. 2020, 2021].

Let $\mathbb{R}[\mathbf{u}] = \mathbb{R}[u_1, ..., u_k]$ be the ring of polynomials in $\mathbf{u}$ and $\mathbb{R}[\mathbf{u}]_d \subset \mathbb{R}[\mathbf{u}]$ be the subset of polynomials of maximum degree d.

Σ_d is the set of SOS polynomials of maximum degree d, defined as

$$\Sigma_d := \left\{ \sum_i p_i^2 : p_i \in \mathbb{R}[\mathbf{u}]_{\lfloor \frac{d}{2} \rfloor} \right\}. \tag{1}$$

Omitting the subscript d implies unrestricted degree.

Let the polynomial sets $\mathcal{G}, \mathcal{H} \subset \mathbb{R}[\mathbf{u}]$ define a compact semialgebraic domain

$$\mathbb{D} = \left\{ \mathbf{u} \in \mathbb{R}^k : \forall g \in \mathcal{G}, h \in \mathcal{H},\ g(\mathbf{u}) \geq 0, h(\mathbf{u}) = 0 \right\}. \tag{2}$$

Since $\mathcal{G}, \mathcal{H}$ specify $\mathbb{D}$ via inequality and equality constraints, we will sometimes refer to them as inequalities and equalities. Given a polynomial objective function $f(\mathbf{u})$, we can now formulate a generic nonconvex polynomial optimization over the domain $\mathbb{D}$:

$$f^* = f(\mathbf{u}^*) = \min\ f(\mathbf{u}) : \mathbf{u} \in \mathbb{D}. \tag{3}$$

Eq. 3 can be equivalently rewritten as a convex problem with a polynomial positivity constraint

$$f^* = \gamma^* = \max\{\gamma : f(\mathbf{u}) - \gamma \text{ is positive for } \mathbf{u} \in \mathbb{D}\}. \tag{4}$$

γ is constrained in Eq. 4 to be a lower bound for $f(\mathbf{u})$ on $\mathbb{D}$. γ^* is then a maximum lower bound and thus achieves the solution to Eq. 3.

Unfortunately, polynomial positivity constraints are NP-hard to maintain [Parrilo 2000]. In contrast, constraining a polynomial to be SOS—a sufficient condition for positivity–only requires a semidefinite matrix constraint. SOSP leverages this sufficiency to parameterize a subset of polynomials that are positive over $\mathbb{D}$ via the *d-truncated quadratic module*:

$$Q(\mathcal{G}, \mathcal{H})_d = \left\{ s_0 + \sum_{g \in \mathcal{G}} s_g g + \sum_{h \in \mathcal{H}} p_h h : \begin{matrix} s_0 \in \Sigma,\ s_g \in \Sigma_d, \\ p_h \in \mathbb{R}[\mathbf{u}]_d \end{matrix} \right\}. \tag{5}$$

s_g and p_h are known as *multiplier polynomials*, as opposed to g and h which define $\mathbb{D}$. Membership in the module is sufficient to assure positivity over $\mathbb{D}$, allowing Eq. 3 to be relaxed into the convex SDP

$$f_d^* = \max\{\gamma \in \mathbb{R} : f(\mathbf{u}) - \gamma \in Q(\mathcal{G}, \mathcal{H})_d\}. \tag{6}$$

Fortuitously, this sufficient condition is also frequently necessary, as is formalized in the following theorem [Putinar 1993]:

Theorem 3.1 (Putinar's Positivstellensatz). *Let $\mathbb{D} = \{\boldsymbol{u} \in \mathbb{R}^k : \forall g \in \mathcal{G}, h \in \mathcal{H}, g(\boldsymbol{u}) \geq 0, h(\boldsymbol{u}) = 0\}$ be a domain with an algebraic certificate of compactness. Any polynomial $f(\boldsymbol{u})$ that is strictly positive on $\mathbb{D}$ is an element of $Q(\mathcal{G}, \mathcal{H})$.*

Thus $f_d^* \leq f^*$ and under the conditions of Theorem 3.1, $f_d^* \to f^*$ as $d \to \infty$. In practice, many problems exhibit $f^* = f_d^*$ for finite d [Laurent 2007; Marschner et al. 2020, 2021; Nie 2014].

We omit the technical details regarding the dual problem to Eq. 6 and only note the most critical point that if $f_d^* = f^*$ and the dual SDP matrix has rank 1, then the unique $\mathbf{u}^*$ can be read off its first column. This is detected by checking that the second largest eigenvalue of the dual SDP matrix, λ_2, equals zero and is known as *exact recovery*. Curiously, Marschner et al. [2021] show that the first column may contain $\mathbf{u}^*$ even without exact recovery. When $\mathbf{u}^*$ is non-unique, exact recovery is impossible. For more detailed examples, see [Marschner et al. 2020, 2021].

3.2 CCD and Surface-Surface Intersection

Marschner et al. [2021] demonstrate the application of SOSP to several geometric kernel problems. Of particular interest to us are the CCD and surface-surface intersection (SSI) problems. We combine both problems under the collision detection umbrella in §4.2. In the SOSP framework, it is simple to formulate these problems for various spline geometries after Bézier extraction.

We recall their formulation of these problems on quadratic and cubic Bézier triangles. Let

$$\boldsymbol{x}(u, v, t) = \sum_i^{n_B} (\mathbf{p}_i + \mathbf{v}_i t) \phi_i(u, v) \tag{7}$$

denote the embedding of a Bézier triangle with control points $\mathbf{p}_i$, velocities $\mathbf{v}_i$, and n_B basis functions ϕ_i of degree d_ϕ. u and v are coordinates in the pre-image of the Bézier triangle, i.e., a flat triangular base domain. Following Eq. 2, the CCD domain is specified by $\mathbf{u} = (u_1, v_1, u_2, v_2, t)$ satisfying certain polynomial inequality and equality constraints. With $\mathcal{G}_1 = \{u_1, v_1, 1 - u_1 - v_1\}$ and $\mathcal{G}_2 = \{u_2, v_2, 1 - u_2 - v_2\}$ specifying two flat triangular domains and $\mathcal{G}_t = \{t, 1 - t\}$ specifying the time interval of interest, $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_t$ specifies the subset of space-time considered in the CCD problem. $\mathcal{H}$ is comprised of three equalities, $\mathcal{H} = \{\boldsymbol{x}_1(u_1, v_1, t)_{xyz} - \boldsymbol{x}_2(u_2, v_2, t)_{xyz}\}$, encoding the collision constraint. Since CCD finds the earliest collision, the objective polynomial is time: $f(\mathbf{u}) = t$. The SSI domain is the set $\mathbf{u} = (u_1, v_1, u_2, v_2)$ satisfying inequalities $\mathcal{G}_1 \cup \mathcal{G}_2$ and three equalities $\mathcal{H} = \{\boldsymbol{x}_1(u_1, v_1, 0)_{xyz} - \boldsymbol{x}_2(u_2, v_2, 0)_{xyz}\}$. Since SSI is a feasibility problem, the objective polynomial is arbitrarily chosen as the x-coordinate of the first Bézier triangle $f(\mathbf{u}) = \boldsymbol{x}_1(u_1, v_1, 0)_x$.

The results in [Marschner et al. 2021] show that multiplier polynomials must be degree 5 (6) for quadratic (cubic) Bézier triangle CCD or SSI to be solved reliably. Despite achieving the correct results, they rarely obtain exact recovery in CCD, meaning they are unable to certify that the obtained collision is the earliest possible. Runtimes average around 25 s and 700 s per CCD problem, for quadratic and cubic triangles respectively. The SSI problem is simpler and has accordingly lower runtimes of 0.78 s and 6.19 s.

3.3 Curved paths using Dual Quaternions

Marschner et al. [2021] only consider CCD problems with linear trajectories; we extend their work to curved paths in §4.4. In particular, we target paths through SE(3), the space of rigid body motions, which we represent by *unit dual quaternions*, $\mathbb{DH}_1$. In this section, we provide relevant background on the use of dual quaternions to represent rigid body motions. For an overview, see [Jia 2013].

A dual quaternion $Q = p + \epsilon q \in \mathbb{DH}$ combines two quaternions p and q with the dual unit ϵ, which satisfies the property $\epsilon^2 = 0$. Key to our work is the conjugation operation $Q^\dagger = p^\star + \epsilon q^\star$, where $\star$ denotes the quaternion conjugate. Using this operation, unit dual quaternions are defined as

$$\mathbb{DH}_1 = \{Q \in \mathbb{DH} : Q^\dagger \otimes Q = 1\} = \left\{ p + \epsilon q \in \mathbb{DH} : \begin{matrix} p \cdot q = 0 \\ \|p\| = 1 \end{matrix} \right\} \tag{8}$$

with $\otimes$ denoting dual quaternion multiplication, $\|p\|$ the quaternion norm, and $p \cdot q$ the elementwise quaternion dot product.

Dual quaternions are particularly well suited to our work because many operations, such as transforming a point by a dual quaternion, result in low-degree polynomials perfect for use with SOSP. Given a dual quaternion Q and a point v encoded as a dual quaternion $Q_v = 1 + \epsilon v$, the rigidly transformed point after applying Q is

$$Q_{v'} = Q \otimes Q_v \otimes Q^\dagger, \tag{9}$$

a polynomial in the coordinates of v.

Constructing a path between two rigid transforms boils down to interpolation on the manifold associated with $\mathbb{DH}_1$. Naïve linear interpolation (LERP) of dual quaternions is invalid, as it leaves the $\mathbb{DH}_1 \subset \mathbb{DH}$ manifold. Dual quaternion linear interpolation (DQLERP) solves this by projecting LERP onto $\mathbb{DH}_1$ via normalization [Kavan et al. 2008, § 3.2]:

$$\text{DQLERP}(Q_0, Q_1, t) \coloneqq \frac{Q_0 + t(Q_1 - Q_0)}{\|Q_0 + t(Q_1 - Q_0)\|}. \tag{10}$$

While this interpolation is algebraic and stays on $\mathbb{DH}_1$, it does not respect the underlying geometry of SE(3).

A more canonical interpolation is derived from *screw motions*. Chasles' theorem [Selig 2005] states that all rigid motions are screw motions, i.e., a rotation around a *screw axis* followed by a translation along the same axis. This motion is represented by *screw parameters*: two vectors $(\boldsymbol{l}, \boldsymbol{m})$ denoting the Plücker coordinates of the screw axis, as well as θ and δ representing the rotation angle and translation distance. Plücker coordinates satisfy the conditions $\|\boldsymbol{l}\| = 1$ and $\boldsymbol{l} \cdot \boldsymbol{m} = 0$. Screw parameters $(\boldsymbol{l}, \boldsymbol{m}, \theta, \delta)$ parameterize the Lie algebra $\mathfrak{se}(3)$, the tangent space to SE(3). The exponential map $\exp : \mathfrak{se}(3) \to \text{SE}(3)$ using screw parameters for $\mathfrak{se}(3)$ and $\mathbb{DH}_1$ for SE(3) is [Daniilidis 1999, Eq. 27]

$$\begin{aligned} \exp : (\boldsymbol{l}, \boldsymbol{m}, \theta, \delta) \mapsto \left(c + s\boldsymbol{l} + \epsilon\left(-\frac{\delta s}{2} + \frac{\delta c}{2}\boldsymbol{l} + s\boldsymbol{m}\right)\right) \\ c \coloneqq \cos(\theta/2)\,,\ s \coloneqq \sin(\theta/2)\,. \end{aligned} \tag{11}$$

Using the standard $\log = \exp^{-1}$ map and $Q^\star = p^\star - \epsilon q^\star$, the screw space linear interpolation (ScLERP) [Kavan et al. 2006] is

$$(\boldsymbol{l}, \boldsymbol{m}, \theta, \delta) \coloneqq \log(Q_0^\star Q_1) \tag{12}$$

$$\text{ScLERP}(Q_0, Q_1, t) \coloneqq Q_0 \exp(\boldsymbol{l}, \boldsymbol{m}, t\theta, t\delta). \tag{13}$$

Since Eq. 13 shows rotation angle and translation distance vary linearly in time, angular and translational velocity are constant along the path, a property missing from DQLERP. ScLERP additionally traces a geodesic from Q_0 to Q_1 in SE(3).

4 METHOD

We begin by detailing our speedup methods in § 4.1. Then § 4.2 presents our SOSP collision detection setup. In § 4.3, we introduce an algebraic formulation for the tapered cubic cylinder, and § 4.4 introduces our algebraic expressions for rigid body motions.

4.1 Speeding up SOS

The SOSP framework is flexible but has significant runtime costs. Here, we present several methods for decreasing runtime.

4.1.1 Mixed Degree. The runtime of an SOSP problem depends heavily on the number of monomials in the parameters of the quadratic module, i.e., monomials of s_0, s_g, p_h in Eq. 5. This number grows factorially with degree d. On the other hand, d controls the expressiveness of the quadratic module—a larger d results in a larger module, allowing Eq. 6 to approach Eq. 4. We want the module to be just barely expressive enough to acheive exact recovery. To that end, we introduce the (d_1, d_2)-*truncated quadratic module*

$$\mathcal{Q}(\mathcal{G}, \mathcal{H})_{d_1,d_2} = \left\{ s_0 + \sum_{g \in \mathcal{G}} s_g g + \sum_{h \in \mathcal{H}} p_h h : \begin{matrix} s_0 \in \Sigma,\ s_g \in \Sigma_{d_1} \\ p_h \in \mathbb{R}[\mathbf{u}]_{d_2} \end{matrix} \right\}, \tag{14}$$

which gives us finer control over the expressiveness of the module. Eq. 14 separates the degree truncation of inequality and equality multipliers s_g and p_h, allowing $\deg(p_h h)$ to change independently from $\deg(s_g g)$. In the context of CCD, $\deg(h)$ is typically greater than $\deg(g)$ so choosing $d_1 = d_2$ results in $\deg(p_h h) > \deg(s_g g)$ and a total runtime that is heavily dependent on $\deg(p_h h)$. In this case, changing d_1 and d_2 independently allows us to maintain the expressiveness of s_g, while decreasing the size of p_h and thus both the overall size and runtime dramatically.

Aside from runtime, Marschner et al. [2021] hypothesize that redundant monomials in the problem can prevent exact recovery while still allowing extraction of f^* and $\mathbf{u}^*$. Indeed, our results show that discarding unnecessary monomials from p_h grants us exact recovery where [Marschner et al. 2021] falls short.

4.1.2 Higher-Degree Descriptions of Equivalent Domains. The quadratic module is not directly dependent on domain $\mathbb{D}$ but instead is constructed from the polynomial sets $\mathcal{G}, \mathcal{H}$ used to describe $\mathbb{D}$. This is because different ways of writing the same domain have a significant effect on what functions the quadratic module can express. Additionally, no particular choice of $\mathcal{G}, \mathcal{H}$ is canonical. We observe that it is frequently beneficial for runtime to replace pairs of elements of $\mathcal{G}$ with their product. This produces an alternate description of the same $\mathbb{D}$, which generates a different quadratic module.

For example, consider the interval $\mathbb{D} = \{t : t \in [0, 1]\}$. $\mathbb{D}$ can be encoded equivalently with either two linear polynomials, $\mathcal{G}_{t_1} = \{t, 1 - t\}$, or one quadratic polynomial, $\mathcal{G}_{t_2} = \{t(1 - t)\}$. This choice alters $\sum s_g g$ in Eq. 14; For a fixed multiplier polynomial degree, substituting $\mathcal{G}_{t_1}$ with $\mathcal{G}_{t_2}$ will increase the degree of $\sum s_g g$, but decrease the number of multiplier polynomials used. The choice of module that minimizes runtime depends on the problem structure and the experimentally determined degrees of the multiplier polynomials. We remark on the theoretical connection between rewritings of the module to another variant of the Positivstellensatz in §6.1 of the supplemental materials.

4.1.3 Reducing Degree with Increased Variables. The next speedup approach is to decrease the degree of polynomials in $\mathcal{G}$ by increasing the number of variables and constraints. Consider the toy example:

$$\mathbb{D}_0 = \{x : x^6 \geq 0\}, \quad \mathbb{D}_1 = \{(x, y) : y^2 \geq 0 \text{ and } x - y^3 = 0\}. \tag{15}$$

Projecting $\mathbb{D}_1$ onto its first dimension produces $\mathbb{D}_0$. However, by introducing the new variable y, the maximum degree of its defining polynomials is reduced to 3. Such a decrease has a large impact on the maximum degree of an SOS problem involving $\mathbb{D}_0$. This may seem contrary to § 4.1.2, but in practice reducing runtime is a balance between number of polynomials and degree. It is not obvious when to take this approach, but we show some successful examples in § 5.3 and § 5.4.

4.1.4 Constraining γ. The last speedup method is to constrain the lower bound variable γ from Eq. 6:

$$t_d^* = \max\{\gamma \in [0,1] : t - \gamma \in \mathcal{Q}(\mathcal{G},\mathcal{H})_d\}. \tag{16}$$

This applies specifically to CCD where a min/max time, usually 0/1, is known. This minor adjustment is nothing remarkable, but it is worth noting that $\gamma \in [0,1]$ is a separate constraint from the $t \in [0,1]$ encoded in $\mathcal{G}$.

4.2 SOS Collision Detection (CD) Certificates

Solutions to CD can possess the following certificates: Intersecting Pair (IP), Earliest Collision (EC), or Non-Collision (NC). First, if a pair of geometries already intersects at $t = 0$, we call them intersecting pairs. Only if they do not already intersect do we try to find the earliest time of collision on their trajectory. The non-collision certificate straightforwardly certifies that the pair of geometries do not intersect on their trajectories in the time interval $[0,1]$. We detail the algebraic versions of these certificates and how to obtain them in the notation of §3.2. Minor differences when extending to other geometries are detailed in §5.

A certificate of collision is a point $\mathbf{u} = [u_1, v_1, u_2, v_2, t]$ satisfying $\forall g \in \mathcal{G}, g(\mathbf{u}) \geq 0$ such that

$$\boldsymbol{x}_1(u_1, v_1, t) = \boldsymbol{x}_2(u_2, v_2, t). \tag{17}$$

Earliest Collision (EC) is certified by $\mathbf{u}$ satisfying Eq. 17, which guarantees collision, in conjunction with exact recovery of the CCD problem. Exact recovery certifies that it is the *earliest* collision, and is checked by ensuring the the second eigenvalue, λ_2, equals 0. A problem occurs, however, if the configuration given to a CCD problem is already intersecting at $t = 0$. In this case, two surfaces will generically intersect at more than one location, making exact recovery impossible. The next certificate resolves this.

Intersecting Pair (IP) is certified by $\mathbf{u}$ satisfying Eq. 17 with $t = 0$. This is obtained by solving the SSI problem in §3.2. Since the objective polynomial of the SSI problem is not time, a unique solution is generically obtained. Going forward, we consider collision detection on geometries of different dimensions so we will refer to the surface specific SSI as the more general IP problem.

Non-Collision (NC) is certified by a positive value γ and truncation degree d satisfying

$$D^2(\mathbf{u}) - \gamma \in \mathcal{Q}(\mathcal{G},\emptyset)_d, \tag{18}$$

where $D^2(\mathbf{u}) = \|\boldsymbol{x}_1(u_1, v_1, t) - \boldsymbol{x}_2(u_2, v_2, t)\|^2$ is the squared distance between geometries. Note that maximizing γ subject to Eq. 18 is exactly the SOS formulation of minimizing $D^2(\mathbf{u})$ over $\mathbb{D}$. A key difference however is that any positive γ suffices as a non-collision certificate. As a result, unlike the CCD case, exact recovery is unnecessary to certify non-collision. Additionally, we can truncate the quadratic module more heavily than if γ needed to be a tight bound, leading to a smaller overall optimization problem. If we do happen to use a d large enough for exact recovery, then $\sqrt{\gamma}$ is a certifiable minimum separation distance.

Numerical Issues. Since we solve SDPs with [MOSEK ApS 2020] using finite precision, equations like $\lambda_2 = 0$, $\gamma > 0$, and Eq. 17 can only be verified within a certain tolerance. With these tolerances, the effective algebraic certificates are:

$$\lambda_2 \leq \epsilon_\lambda, \quad \gamma \geq \epsilon_\gamma, \quad D(\mathbf{u}) \leq \epsilon_x. \tag{19}$$

These tolerances introduce the possibility of failing to certify any of the listed cases algebraically. For example, if two non-colliding Bézier triangles skim past each other with a minimum squared separation distance of 10^{-9}, and the non-collision tolerance was chosen as $\epsilon_\gamma = 10^{-8}$, it will be impossible to certify non-collision. On the other hand, $\epsilon_\gamma = 10^{-10}$ pushes the limits of finite precision solvers like [MOSEK ApS 2020], i.e., it is possible to convince SDP solvers that a polynomial $x^2 - 10^{-10}$ is non-negative. We take a conservative approach, making sure that collision and non-collision are never certified at the same time. The few remaining cases are labeled *Inconclusive* (I). Further discussion is included in supplemental materials.

4.3 Tapered Cubic Cylinder (TCC)

Motivated by the fact that production hair pipelines use cubic splines for modeling and rendering but convert to linear segments for simulation [Thyng et al. 2017], we algebraically formulate a suitable geometry for curved hair bundle collision detection: the *tapered cubic cylinder*.

A TCC is essentially a cubic spline centerline with a radius varying linearly from l_0 to l_1. Algebraically, a TCC is

$$\mathbb{T} = \left\{ c(\xi,t) + [x,y,z]^T : \begin{matrix} x^2+y^2+z^2 = ((1-\xi)l_0 + \xi l_1)^2 \\ \text{and } \xi, t \in [0,1]^2 \end{matrix} \right\}, \tag{20}$$

where $c(\xi, t)$ is a time-varying cubic Bézier curve (cubic in ξ, linear in t). Though stated as a cylinder, a TCC includes spherical endcaps. §5.3 shows results on collision detection for this geometry.

4.4 CCD on Dual Quaternion Paths

4.4.1 Algebraic Path Formulations. As discussed in § 3.3, rigid trajectories can be constructed by interpolating dual quaternions. For SOSP to be applicable, though, these interpolations must be algebraic. DQLERP applied to a point combines Eq. 9 and Eq. 10 producing a rational expression compatible with SOSP. The more natural ScLERP path unfortunately includes trigonometric functions in Eq. 11. To remedy this non-algebraicness, we introduce a novel interpolation method, $\text{ScLERP}_{\text{Tay}}$, which approximates ScLERP qualitatively and quantitatively. This is achieved by replacing $\sin(\frac{1}{2}\theta t)$ and $\cos(\frac{1}{2}\theta t)$ in the exp function Eq. 11 with their Taylor approximations in t from the base point x_0. Applying Eq. 13 with this modified exp function produces an interpolation method we denote $\text{ScLERP}_{\text{Tay},x_0}$. This trajectory converges to ScLERP as the degree of the Taylor approximation increases, but only exactly equals ScLERP at x_0. Since both the beginning and end points of the interpolation are expected to adhere exactly to the input, we construct two Taylor approximations centered about the beginning and end points, and linearly blend between them. Finally, since the approximation creates deviation from $\mathbb{DH}_1$ we renormalize:

$$Q_t = t \cdot \text{ScLERP}_{\text{Tay},0} + (1-t) \cdot \text{ScLERP}_{\text{Tay},1} \tag{21}$$

$$\text{ScLERP}_{\text{Tay}} = \frac{Q_t}{\|\text{real}(Q_t)\|}. \tag{22}$$

The resulting interpolation scheme, $\text{ScLERP}_{\text{Tay}}$, produces a rational expression for the trajectory and approximates ScLERP. The denominator of Eq. 22 is simpler than a general dual quaternion norm thanks to the structure of screw space transformations. Recall from

Eq. 8 that a unit dual quaternion $Q = p + \epsilon q$ satisfies $\|p\| = 1$ and $p \cdot q = 0$. In this case, $p \cdot q$ for Q_t always yields 0, a fact derived from the properties $\|\boldsymbol{l}\| = 1$ and $\boldsymbol{l} \cdot \boldsymbol{m} = 0$ of the screw parameters. This simplification drastically decreases the overall degree of $\text{ScLERP}_{\text{Tay}}$ paths, making them more tractable in CCD problems.

Fig. 2 shows a comparison of $\mathbb{DH}_1$ interpolation schemes. Even with only a degree 1 Taylor approximation, $\text{ScLERP}_{\text{Tay}}$ is dramatically closer both quantitatively and qualitatively to the ScLERP path than the DQLERP path is.

Another consideration when selecting an interpolation method is its polynomial degree. Applying the interpolated quaternion to a point results in a rational polynomial

$$Q_{v_t} := Q_t \otimes Q_v \otimes Q_t^\dagger := \frac{N_{v_t}}{D_{v_t}} \tag{23}$$

with a polynomial dual quaternion numerator, N_{v_t}, and a polynomial scalar denominator, D_{v_t}. The degree of the path is the degree of the time variable t in Eq. 23; these degrees are shown for the algebraic interpolation schemes we consider in Fig. 2. Since the degree of the path is correlated with the runtime of the final SOSP problem, DQLERP's lower degree makes it sometimes advantageous over $\text{ScLERP}_{\text{Tay}}$ as it can make complex formulations feasible in a more reasonable runtime.

4.4.2 Writing the CCD Problem. We now formulate the CCD problem for two objects with polynomial geometry following dual quaternion paths using SOSP. We also consider the simpler problem of colliding a curved object with a stationary plane. These formulations can be straightforwardly generalized to any geometry expressible as an algebraic map $\boldsymbol{P}(\mathbf{u})$ over a base domain.

The domain of the SOSP problem is specified by inequality constraints for the time interval and the geometric object as described in § 3.2. Applying Eq. 23 with $Q_v = 1 + \epsilon \boldsymbol{P}(\mathbf{u})$ gives a rational polynomial representation of every point on the object under rigid motion. For CCD between a rigidly moving object and a stationary plane defined by normal $\boldsymbol{n}$ and offset α, collision is encoded by one equality constraint

$$\boldsymbol{n} \cdot \text{dual}\left\{N_{v_t}\right\} - \alpha D_{v_t} = 0, \tag{24}$$

where dual$\{\cdot\}$ extracts the vector part of the dual part of a dual quaternion, which for a dual quaternion representing a point extracts the $\mathbb{R}^3$ coordinates of that point. For the CCD problem between two rigidly moving objects, there are three collision equality constraints defined by equating points on the two objects and cross-multiplying:

$$\text{dual}\left\{N_{v_t,A}\right\} D_{v_t,B} - \text{dual}\left\{N_{v_t,B}\right\} D_{v_t,A} = 0. \tag{25}$$

Curved geometry and trajectory CCD ends up being the largest problem we solve, leading us to use minimal bounding ellipsoids from [Marschner et al. 2021] as a culling step. The CCD problem for two ellipsoids moving along dual quaternion paths is its own SOSP. Ellipsoids can be encoded as a dual quaternion rotation Q_e and three axis lengths $\mathcal{L} \in \mathbb{R}^3$. Let a time interval and two balls be specified by variables $\{t, \boldsymbol{x}_A, \boldsymbol{x}_B\}$, with $\boldsymbol{x}_A, \boldsymbol{x}_B \in \mathbb{R}^3$, and inequalities $\mathcal{G} = \{t, 1-t, 1-\|\boldsymbol{x}_A\|^2, 1-\|\boldsymbol{x}_B\|^2\}$. We apply Eq. 23 with $Q_{v_A} = 1+\epsilon(\boldsymbol{x}_A \odot \mathcal{L}_A)$ (using the elemetwise product $\odot$) and $Q_{t_A} = \text{interp}(Q_0, Q_1, t) \otimes Q_{e_A}$, where interp is the desired $\mathbb{DH}_1$ interpolation, to encode a rigidly moving ellipsoid. This yields $N_{v_t,A}$ and $D_{v_t,A}$, which along with the analogous result for the second ellipsoid, allow us to construct the collision equality constraint from Eq. 25.

5 RESULTS

To illustrate the effect of our speedups, §5.1 and §5.2 present results for curve and triangle collision detection, the latter of which is compared to [Marschner et al. 2021]. Enabled by these speedups, we apply our collision detection formulation to the tapered cubic cylinder in §5.3 and demonstrate the solution of rigid body motion CCD problems using SOSP in §5.4. Our SOSP problems require the choice of degree of the multiplier polynomials. We report the degrees we find empirically necessary to solve each problem in this section and in the supplemental tables. We refer the reader to [Marschner et al. 2021, §5] for further discussion on the choice of SOSP degree. Our runtimes are obtained using MATLAB 2021a, YALMIP v20200116 and MOSEK 9.2, with the rigid body motion problems running on a 2.8 GHz 4-Core Intel i7 CPU and all other tests on a 3.7 GHz Intel i7-8700K CPU. At a high level, our new formulations provide up to a 338× speedup. A more detailed breakdown is provided in Table 1 and Fig. 5. Since YALMIP time is not incurred per collision problem, we only report runtimes from the MOSEK solver.

Since this section formulates many SOS programs, we label them based on the problem they solve (CCD, IP, NC), subscripted with the truncation degrees of their quadratic modules (d or d_1, d_2), and superscripted with geometry type (1D for curves, 2D for triangles, or TCC). We reference these problems in supplemental Tables 1 and 2 with specific truncation degrees subbed in. Expanded descriptions of all SOSPs are provided in the supplementary materials.

5.1 Speeding up CD on Bézier Curves

CCD for Bézier Curves. Let $\mathbf{c}_1(\xi, t) = \sum_i^{n_B} (\mathbf{p}_i + t\mathbf{v}_i)\phi_i(\xi)$ and $\mathbf{c}_2(\zeta, t) = \sum_i^{n_B} (\mathbf{q}_i + t\mathbf{w}_i)\phi_i(\zeta)$ denote two Bézier curves with basis functions ϕ_i of degree $d_\phi = \{1, 2, 3\}$. The basic CCD problem following [Marschner et al. 2021] is specified with polynomial sets $\mathcal{G}_a = \{\xi, 1-\xi, \zeta, 1-\zeta, t, 1-t\}$, $\mathcal{H} = \{\mathbf{c}_1(\xi, t) - \mathbf{c}_2(\zeta, t)\}$.

$$\max_{\gamma \in \mathbb{R}} \gamma \ : \ t - \gamma \in Q(\mathcal{G}_a, \mathcal{H})_d \tag{CCD_d^{1D}}$$

Combining the speedups in § 4.1.1, § 4.1.2, and § 4.1.4, we arrive at the alternative formulation specified with polynomials $\mathcal{G}_b = \{\xi(1-\xi), \zeta(1-\zeta), t(1-t)\}$:

$$\max_{\gamma \in [0,1]} \gamma : \ t - \gamma \in Q(\mathcal{G}_b, \mathcal{H})_{d_1,d_2}. \tag{$\text{CCD}_{d_1,d_2}^{1D}$}$$

Modifications made to the basic formulation in CCD_d^{1D} to obtain the formulation $\text{CCD}_{d_1,d_2}^{1D}$ are highlighted in red.

NC for Bézier Curves. Since static Bézier curves do not generically intersect, we do not consider the IP problem. On the non-collision side using speedups from § 4.1.2 and § 4.1.4 we get

$$\max_{\gamma \geq 0} \gamma \ : \ \|\mathbf{c}_1(\xi, t) - \mathbf{c}_2(\zeta, t)\|_2^2 - \gamma \in Q(\mathcal{G}_b, \emptyset)_d. \tag{NC_d^{1D}}$$

Since there are no equality constraints, we do not use mixed degree. Any solution to NC_d^{1D} is a lower bound to separation distance, but if we also obtain exact recovery, the bound is tight.

We perform a batch test on 1000 configurations of $\{\mathbf{p}_i, \mathbf{q}_i, \mathbf{v}_i, \mathbf{w}_i\}$ randomly generated according to a standard normal distribution. We then uniformly sample a directional vector $\boldsymbol{d}$ of length $\frac{1}{3}$ and

Table 1: Batch CD results for curves and triangles of degree $d_\phi = \{1, 2, 3\}$. Average runtimes are provided for CCD using our new speedups, compared to [Marschner et al. 2021] without culling strategies. Our improvements produce significant speedups in all cases. Additionally, culling gives even greater speedups by certifying non-collision or intersecting-pair and thus allowing us to skip trying to certify earliest-collision with CCD. Culling is especially effective for larger problems.

		Status				Time w/o culling (ms)			Time w/ culling (ms)	
	d_ϕ	IP	EC	NC	I	Old	New	Speedup	New	Speedup
	1	-	115	885	0	4.2	3.0	1.4x	6.9	0.62x
Curves	2	-	216	784	0	43	4.1	10x	9.0	4.8x
	3	-	691	308	1	669	127	5.3x	25	26x
	1	261	295	442	2	798	204	3.9x	21	38x
Triangles	2	702	260	35	3	22571	913	25x	430	53x
	3	393	531	76	0	$7.0 \cdot 10^5$	32581	21x	2070	338x

translate $\mathbf{p}_i, \mathbf{w}_i$ by $\boldsymbol{d}$ and $\mathbf{q}_i, \mathbf{v}_i$ by $-\boldsymbol{d}$. This shifting step encourages collisions between splines $\mathbf{c}_1$ and $\mathbf{c}_2$. We provide a detailed breakdown of our culling strategy in supplemental Table 1 and compare to CCD from [Marschner et al. 2021]. Additionally, Fig. 3 illustrates the relative impacts of speedups applied in different combinations. Tolerances are $\epsilon_\lambda = \epsilon_x = 10^{-4}, \epsilon_\gamma = 10^{-6}$. Our improvements show speedups in Table 1 as high as 26×. The only slow-down comes from aggregated runtime of linear CD because, unlike [Marschner et al. 2021], our new method provides a certificate for non-colliding cases.

The only method we are aware of targeting collision detection in as much generality as we do is [Snyder 1992] and its followup [Snyder et al. 1993] to handle multi-point collisions. Wang et al. [2021] provide a modern implementation of [Snyder 1992] for linear segment CCD, resulting in an average runtime of .0124 seconds per collision on a 2.35 GHz AMD EPYC 7452. Though not directly comparable, our linear segment CCD runtime averages .0030 per collision, which is on par with the method of Wang et al. [2021]. On their challenging benchmark dataset for linear segment CCD, we report 269 false positives compared to the 214 from [Wang et al. 2021] and 141 from [Snyder 1992]. All methods report 0 false negatives.

5.2 Speeding up CD on Bézier Triangles

We compare our runtimes to [Marschner et al. 2021] where a single d is chosen large enough to detect collision, and is then applied uniformly to the batch.

CCD for Bézier Triangles. To align with our equation label format, the old Bézier triangle CCD formulation described in §3.2 will be referred to as CCD_d^{2D} going forward. Using the same notation, we introduce two modified CCD formulations

$$\max_{\gamma\in[0,1]} \gamma \ : \ t-\gamma \in Q(\mathcal{G}_1 \cup \mathcal{G}_2 \cup \{t(1-t)\}, \mathcal{H})_{d_1,d_2} \quad (\text{CCD}_{d_1,d_2}^{2Da})$$

$$\max_{\gamma\in[0,1]} \gamma \ : \ t-\gamma \in Q(\mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_t, \mathcal{H})_{d_1,d_2} \quad (\text{CCD}_{d_1,d_2}^{2Db})$$

with the difference between $\text{CCD}_{d_1,d_2}^{2Da}$ and $\text{CCD}_{d_1,d_2}^{2Db}$ highlighted.

IP for Bézier Triangles. The IP formulation is

$$\max_{\gamma\in\mathbb{R}} \gamma \ : \ \boldsymbol{x}_1(u_1, v_1, 0)_x - \gamma \in Q(\mathcal{G}_1 \cup \mathcal{G}_2, \mathcal{H})_{d_1,d_2} \quad (\text{IP}_{d_1,d_2}^{2D})$$

The IP objective has notably asymmetric dependence on $\boldsymbol{x}_1$. For symmetric coverage, if the first application of the IP problem fails to return a certificate, we apply the same IP problem again with swapped parameters $\mathbf{p}_i \leftrightarrow \mathbf{q}_i, \mathbf{v}_i \leftrightarrow \mathbf{w}_i$.

NC for Bézier Triangles. Lastly, the sped up NC problem is

$$\max_{\gamma\geq 0} \gamma : D^2(\mathbf{u}) - \gamma \in Q(\mathcal{G}_1 \cup \mathcal{G}_2 \cup \{t(1-t)\}, \emptyset)_d. \quad (\text{NC}_d^{2D})$$

Again, we generate 1000 random $\{\mathbf{p}_i, \mathbf{q}_i, \mathbf{v}_i, \mathbf{w}_i\}$ configurations as in § 5.1, with the modification that the length of $\boldsymbol{d}$ is $\frac{1}{3}, \frac{1}{2}, \frac{1}{1.625}$ for $d_\phi = 1, 2, 3$ respectively. We choose these lengths to roughly balance the number of IP and CCD cases.

Supplemental Table 1 shows a detailed breakdown of our culling strategy compared to formulations from [Marschner et al. 2021]. Tolerances are $\epsilon_\lambda = \epsilon_x = 10^{-4}, \epsilon_\gamma = 10^{-7}$ for flat and quadratic triangles and $\epsilon_\lambda = \epsilon_x = 10^{-3}, \epsilon_\gamma = 10^{-6}$ for cubic triangles. Speedups are summarized in the first three rows of Table 1 and are as high as 338×.

Inconclusive Cases. The one inconclusive case from cubic spline CD is correctly classified as non-collision if $\epsilon_\gamma = 5 \cdot 10^{-7}$. Similarly, the two inconclusive flat triangle CD cases can be correctly certified as non-collision with $\epsilon_\gamma = 3 \cdot 10^{-8}$. The 3 inconclusive cases from quadratic triangle CD can be correctly classified as collisions if $\epsilon_\lambda = \epsilon_x = 2 \cdot 10^{-3}$. These parameter choices would still maintain that no cases are double-certified but would require careful selection based on empirical data. We expect that in practice inconclusive cases are inevitable and leave them for a robust collision response method to handle.

5.3 CD on TCC

Our speedups allow us to tackle more complex problems like collision detection on TCC. Using notation from §5.1, two time-varying TCCs are specified with $\{\mathbf{p}_i, \mathbf{q}_i, \mathbf{v}_i, \mathbf{w}_i\}$ indicating parameters of the time-dependent cubic spline centerlines $\mathbf{c}_1(\xi, t), \mathbf{c}_2(\zeta, t)$. The first (second) TCC has radii interpolating linearly from l_0 (m_0) to l_1 (m_1) w.r.t. ξ (ζ). Let $\mathcal{G}_{b_1} = \{\xi, 1-\xi, \zeta, 1-\zeta\}$, $\mathcal{G}_{b_2} = \{\xi(1-\xi), \zeta(1-\zeta)\}$, $\mathcal{G}_c = \{t(1-t)\}$ be the inequality sets constraining $(\xi, \zeta, t) \in [0, 1]^3$. From Eq. 20 we read off equalities parameterizing two spheres of linearly varying radius

$$\mathcal{H}_1 = \begin{Bmatrix} x_1^2 + y_1^2 + z_1^2 - ((1-\xi)l_0 + \xi l_1)^2, \\ x_2^2 + y_2^2 + z_2^2 - ((1-\zeta)m_0 + \zeta m_1)^2 \end{Bmatrix}$$

and collision equalities

$$\mathcal{H}_2 = \{\mathbf{c}_1(\xi,t) + [x_1,y_1,z_1]^T - \mathbf{c}_2(\zeta,t) - [x_2,y_2,z_2]^T\}.$$

Combining these polynomials gives us the CCD problem:

$$\max_{\gamma\in[0,1]} \gamma \;:\; t-\gamma \in \mathcal{Q}(\mathcal{G}_{b_2}\cup\mathcal{G}_c, \mathcal{H}_1\cup\mathcal{H}_2)_{d_1,d_2}. \qquad (\mathrm{CCD}^{TCC}_{d_1,d_2})$$

Non-collision is certified by ensuring spline centerlines have a separation distance larger than the sum of their radii. Let $D^2(\xi,\zeta,t) = \|\mathbf{c}_1(\xi,t) - \mathbf{c}_2(\zeta,t)\|_2^2$ be the squared distance between centerlines and $r^2(\xi,\zeta) = ((1-\xi)l_0+\xi l_1+(1-\zeta)m_0+\zeta m_1)^2$ be the squared sum of TCC radii. The NC problem is then just a feasibility problem—no optimization variable is necessary:

$$D^2(\xi,\zeta,t) - r^2(\xi,\zeta) \in \mathcal{Q}(\mathcal{G}_{b_2}\cup\mathcal{G}_c,\emptyset)_d. \qquad (\mathrm{NC}^{TCC}_{d})$$

Finally, we use two different IP formulations.

$$\max_{\gamma\in\mathbb{R}} \gamma \;:\; D^2(\xi,\zeta,0) - r^2(\xi,\zeta) - \gamma \in \mathcal{Q}(\mathcal{G}_{b_1},\emptyset)_d \qquad (\mathrm{IP}^{TCCa}_{d})$$

$$\max_{\gamma\in\mathbb{R}} \gamma \;:\; \mathbf{u}^T\vec{r} - \gamma \in \mathcal{Q}(\mathcal{G}_{b_2}\cup\mathcal{G}_c,\mathcal{H}_1\cup\mathcal{H}_2)_{d_1,d_2} \qquad (\mathrm{IP}^{TCCb}_{d_1,d_2})$$

where $\mathbf{u} = [\xi,\zeta,x_1,y_1,z_1,x_2,y_2,z_2]^T$ is a vector of all variables in the problem and $\vec{r}$ is a random vector. $\mathrm{IP}^{TCCb}_{d_1,d_2}$ is an application of §4.1.3 to IP^{TCCa}_{d}: by introducing six variables x_1,y_1,z_1,x_2,y_2,z_2, the objective drops from degree 8 to 1.

As in previous subsections, we perform 1000 random CD tests. Spline centerlines are generated in the same way as §5.1. Radii l,m are sampled uniformly in the range $[0,0.2]+0.01$. Tolerances are $\epsilon_\lambda = \epsilon_x = 10^{-4}$. The NC problem only requires feasibility so no ϵ_γ is needed. We were able to certify answers for all 1000 cases, with 459 NC, 210 IP, and 331 CCD cases. A more detailed breakdown and runtimes are given in supplemental Table 2.

5.4 Rigid Body Motion CCD

The rigid body motion CCD problems described in §4.4 have exceptionally large degrees, making the application of our speedups especially important.

Fig. 1 demonstrates the collision problem between a stationary plane and various polynomial geometries moving along $\mathrm{ScLERP}_{\mathrm{Tay}}$ paths with Taylor approximation degree 1. For each of the geometry types, we acheive a speedup by using a reformulated domain as described in §4.1.2. The largest speedups come from combining pairs of linear domain constraints involving the same variable into a single quadratic constraint. For example, when colliding bicubic Bézier patches, we use the inequalities $\mathcal{G} = \{t(t-1), u(1-u), v(1-v)\}$, where u,v denote coordinates in the unit-square pre-image of the patch. This exact formulation is not applicable to the pre-image of Bézier triangles, so we instead use $\mathcal{G} = \{t(t-1), u, v, 1-u-v\}$. We then apply the speedup method from §4.1.1, and achieve exact recovery on all colliding examples tested with degree $[d_1,d_2] = [2,1]$ for linear triangle collisions and $[d_1,d_2] = [6,2]$ for all other problems. A detailed summary of these tests is presented in supplemental Table 3.

The CCD problem between two rigidly moving bicubic patches is the largest we consider. Even with our speedups applied, it is infeasible to use the $\mathrm{ScLERP}_{\mathrm{Tay}}$ interpolation. Instead we use DQLERP, which lowers the degree of the largest polynomial by 4. Since the runtime of SOSP problems grow factorially with degree, this leads to a large reduction in runtime. Before applying our speedups, the problem on a DQLERP path is also infeasible—it runs out of memory with a largest SDP matrix of size 501×501. Our speedups decrease this size to 158×158. Furthermore, we cull the number of expensive patch-patch CCD problems by first computing CCD between bounding ellipsoids for the two objects. We use ellipsoids over alternatives such as axis aligned bounding boxes (AABBs) since they provide a tighter bounding volume and better demonstrate application of SOSP to collision detection. After applying the bounding volume check to the experiment in Fig. 6, we only need to solve 76 cases of patch-patch CCD, speeding up the overall computation time by 12.7×. For the ellipsoid-ellipsoid CCD problem, we combine the $t, 1-t$ constraints into a quadratic constraint. We also make use of §4.1.3 by introducing new variables y_A and y_B which are constrained to equal the denominators of the interpolated point dual quaternions through equality constraints $\{y_A - D_{v_t,A} = 0,\ y_B - D_{v_t,B} = 0\}$. This changes the structure of the problem, bringing the mixed degree needed to solve it from $[2,3]$ to $[2,2]$ and decreasing the overall problem size. The results of this test are summarized in Fig. 6.

6 CONCLUSION & FUTURE WORK

The speedups presented in this paper make problems formulated in [Marschner et al. 2021] far more tractable. We apply these speedups towards high degree collision detection problems and demonstrate improved guarantees and runtime. We additionally enable collision detection on tapered cubic cylinders and rigid trajectories through a novel dual quaternion interpolation method.

In addition to further speedups, robust collision response methods are clearly necessary for high-order simulation. High-order simulation will also require the ability to prevent self-intersection. One approach may be to constrain the conveniently rational repulsive surfaces energy [Yu et al. 2021]. These challenges merit further study to enable reliable high-order geometry processing.

ACKNOWLEDGMENTS

The MIT Geometric Data Processing group acknowledges the generous support of Army Research Office grants W911NF2010168 and W911NF2110293, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grants IIS-1838071 and CHS-1955697, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, and from a Google Research Scholar award. Paul Zhang acknowledges the generous support of Mathworks Engineering Fellowship and DOE Computational Science Graduate Fellowship.

REFERENCES

Amir Ahmadi, Georgina Hall, Ameesh Makadia, and Vikas Sindhwani. 2017. Geometry of 3D Environments and Sum of Squares Polynomials. In *Robotics: Science and Systems XIII* (Cambridge, Massachusetts), Nora Ayanian Nancy Amato, Siddhartha Srinivasa and Scott Kuindersma (Eds.). 9 pages. https://doi.org/10.15607/RSS.2017.XIII.071

Farid Alizadeh. 1995. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization* 5, 1 (1995), 13–51. https://doi.org/10.1137/0805002

Alexandre Amice, Hongkai Dai, Peter Werner, Annan Zhang, and Russ Tedrake. 2023. Finding and Optimizing Certified, Collision-Free Regions in Configuration Space for Robot Manipulators. In *Algorithmic Foundations of Robotics XV. WAFR 2022*

(Springer Proceedings in Advanced Robotics, Vol. 25). Springer, Cham, 328–348. https://doi.org/10.1007/978-3-031-21090-7_20

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete Viscous Threads. *ACM Transactions on Graphics* 29, 4, Article 116 (jul 2010), 10 pages. https://doi.org/10.1145/1778765.1778853

Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. 2006. Super-helices for predicting the dynamics of natural hair. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 1180–1187.

Grigoriy Blekherman, Pablo A. Parrilo, and Rekha R. Thomas. 2012. *Semidefinite Optimization and Convex Algebraic Geometry*. Society for Industrial and Applied Mathematics, Philadelphia, PA. https://doi.org/10.1137/1.9781611972290

Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. *ACM Transactions on Graphics* 37, 3, Article 33 (jul 2018), 22 pages. https://doi.org/10.1145/3182159

Konstantinos Daniilidis. 1999. Hand-Eye Calibration Using Dual Quaternions. *The International Journal of Robotics Research* 18, 3 (1999), 286–298. https://doi.org/10.1177/02783649922066213

Laura De Lorenzis, Peter Wriggers, and Thomas J.R. Hughes. 2014. Isogeometric contact: a review. *GAMM-Mitteilungen* 37, 1 (2014), 85–123. https://doi.org/10.1002/gamm.201410005

Zachary Ferguson, Pranav Jain, Denis Zorin, Teseo Schneider, and Daniele Panozzo. 2022. High-Order Incremental Potential Contact for Elastodynamic Simulation on Curved Meshes. arXiv:2205.13727 [cs.GR]

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, and Daniele Panozzo. 2021. Intersection-Free Rigid Body Dynamics. *ACM Transactions on Graphics* 40, 4, Article 183 (jul 2021), 16 pages. https://doi.org/10.1145/3450626.3459802

Thomas J. R. Hughes, John Austin Cottrell, and Yuri Bazilevs. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 39 (2005), 4135 – 4195. https://doi.org/10.1016/j.cma.2004.10.008

Yan-Bin Jia. 2013. Dual quaternions. Iowa State University: Ames, IA, USA. https://faculty.sites.iastate.edu/jia/files/inline-files/dual-quaternion.pdf

Ladislav Kavan, Steven Collins, Carol O'Sullivan, and Jiří Žára. 2006. *Dual quaternions for rigid transformation blending*. Technical Report TCD-CS-2006–46. Trinity College Dublin.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2008. Geometric Skinning with Approximate Dual Quaternion Blending. *ACM Transactions on Graphics* 27, 4, Article 105 (nov 2008), 23 pages. https://doi.org/10.1145/1409625.1409627

Christopher Kulla, Alejandro Conty, Clifford Stein, and Larry Gritz. 2018. Sony Pictures Imageworks Arnold. *ACM Transactions on Graphics* 37, 3, Article 29 (aug 2018), 18 pages. https://doi.org/10.1145/3180495

Monique Laurent. 2007. Semidefinite representations for finite varieties. *Mathematical programming* 109, 1 (2007), 1–26. https://doi.org/10.1007/s10107-004-0561-4

Johan Löfberg. 2004. YALMIP: A Toolbox for Modeling and Optimization in MATLAB. In *IEEE International Symposium on Computer Aided Control System Design (CACSD)* (Taipei, Taiwan). IEEE, Piscataway, NJ, 284–289. https://doi.org/10.1109/CACSD.2004.1393890

Jia Lu and Chao Zheng. 2014. Dynamic cloth simulation by isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 268 (2014), 475–493. https://doi.org/10.1016/j.cma.2013.09.016

Zoë Marschner, David Palmer, Paul Zhang, and Justin Solomon. 2020. Hexahedral Mesh Repair via Sum-of-Squares Relaxation. *Computer Graphics Forum* 39, 5 (2020), 133–147. https://doi.org/10.1111/cgf.14074

Zoë Marschner, Paul Zhang, David Palmer, and Justin Solomon. 2021. Sum-of-Squares Geometry Processing. *ACM Transactions on Graphics* 40, 6, Article 253 (dec 2021), 13 pages. https://doi.org/10.1145/3478513.3480551

MOSEK ApS. 2020. The MOSEK optimization toolbox for MATLAB manual. Version 8.1.0.83. https://docs.mosek.com/8.1/toolbox.pdf

Koji Nakamaru and Yoshio Ohno. 2002. Ray tracing for curves primitive. *Journal of WSCG* 10, 1-2 (2002), 6 pages. http://wscg.zcu.cz/wscg2002/Papers_2002/A83.pdf

Yurii Nesterov and Arkadii Nemirovskii. 1994. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Philadelphia, PA. https://doi.org/10.1137/1.9781611970791

Jiawang Nie. 2014. Optimality conditions and finite convergence of Lasserre's hierarchy. *Mathematical programming* 146, 1 (2014), 97–121. https://doi.org/10.1007/s10107-013-0680-x

Pablo A. Parrilo. 2000. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. Ph. D. Dissertation. California Institute of Technology. https://doi.org/10.7907/2K6Y-CH43

Pablo A. Parrilo. 2019. Algebraic Techniques and Semidefinite Optimization. https://learning-modules.mit.edu/materials/index.html?uuid=/course/6/sp19/6.256#materials.

George Vikram Paul. 1997. *Kinematics of Objects in Contact using Dual Vectors and its Applications*. Ph. D. Dissertation. Carnegie Mellon University. https://www.ri.cmu.edu/pub_files/pub3/paul_george_1997_1/paul_george_1997_1.pdf

Xavier Provot. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation'97: Proceedings of the Eurographics Workshop in Budapest, Hungary, September 2–3, 1997*. Springer, 177–189.

Mihai Putinar. 1993. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal* 42, 3 (1993), 969–984. https://www.jstor.org/stable/24897130

Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. 2000. An algebraic solution to the problem of collision detection for rigid polyhedral objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 4. IEEE, Piscataway, NJ, 3733–3738. https://doi.org/10.1109/ROBOT.2000.845313

Teseo Schneider, Yixin Hu, Xifeng Gao, Jérémie Dumas, Denis Zorin, and Daniele Panozzo. 2022. A Large-Scale Comparison of Tetrahedral and Hexahedral Elements for Solving Elliptic PDEs with the Finite Element Method. *ACM Transactions on Graphics* 41, 3, Article 23 (mar 2022), 14 pages. https://doi.org/10.1145/3508372

Jonathan Mark Selig. 2005. *Geometric Fundamentals of Robotics* (2nd ed.). Springer, New York, NY. https://doi.org/10.1007/b138859

John M. Snyder. 1992. Interval Analysis for Computer Graphics. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*. Association for Computing Machinery, New York, NY, USA, 121–130. https://doi.org/10.1145/142920.134024

John M. Snyder, Adam R. Woodbury, Kurt Fleischer, Bena Currin, and Alan H. Barr. 1993. Interval Methods for Multi-Point Collisions between Time-Dependent Curved Surfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 321–334. https://doi.org/10.1145/166117.166158

Marc Thyng, Christopher Evart, Toby Jones, and Aleka McAdams. 2017. The Art and Technology of Hair Simulation in Disney's Moana. In *ACM SIGGRAPH 2017 Talks* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 25, 2 pages. https://doi.org/10.1145/3084363.3085072

Ty Trusty, Honglin Chen, and David I. W. Levin. 2021. The Shape Matching Element Method: Direct Animation of Curved Surface Models. *ACM Transactions on Graphics* 40, 4, Article 69 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459772

Charles W. Wampler and Andrew J. Sommese. 2011. Numerical algebraic geometry and algebraic kinematics. *Acta Numerica* 20 (2011), 469–567. https://doi.org/10.1017/S0962492911000067

Bolun Wang, Zachary Ferguson, Teseo Schneider, Xin Jiang, Marco Attene, and Daniele Panozzo. 2021. A Large-Scale Benchmark and an Inclusion-Based Algorithm for Continuous Collision Detection. *ACM Transactions on Graphics* 40, 5, Article 188 (sep 2021), 16 pages. https://doi.org/10.1145/3460775

Heng Yang, Ling Liang, Luca Carlone, and Kim-Chuan Toh. 2022. An inexact projected gradient method with rounding and lifting by nonlinear programming for solving rank-one semidefinite relaxation of polynomial optimization. *Mathematical Programming* (2022), 1–64. https://doi.org/10.1007/s10107-022-01912-6

Chris Yu, Caleb Brakensiek, Henrik Schumacher, and Keenan Crane. 2021. Repulsive Surfaces. *ACM Transactions on Graphics* 40, 6, Article 268 (dec 2021), 19 pages. https://doi.org/10.1145/3478513.3480521

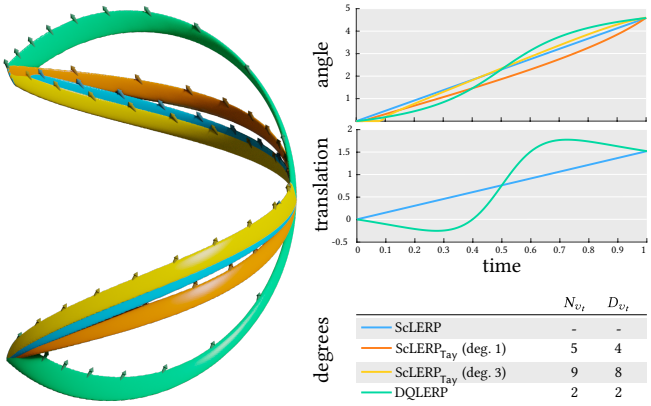


Figure 2: A comparison of the paths generated by different $\mathbb{DH}_1$ interpolation methods, with one direction in the rotation frame plotted at equally spaced t values along the paths. The rotation angle and translation vary linearly over time in the non-polynomial ScLERP path corresponding to a screw motion. The ScLERP$_{\text{Tay}}$ paths using approximation degree 1 and 3 have exactly linearly varying translation components and are both much closer to the ScLERP path than the DQLERP path is. Applying these paths to a point results in Eq. 23 with the shown numerator and denominator degrees.

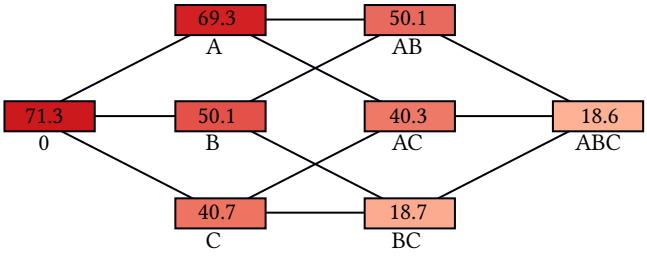


Figure 3: Median runtimes (ms) over 40 cubic spline CCD problems with different speedups applied. Case 0 is CCD_4^{1D}. Case A adds the constraint $\gamma \in [0, 1]$. Case B swaps $\mathcal{G}_a$ for $\mathcal{G}_b$. Case C uses mixed degree $[d_1, d_2] = [4, 3]$. Pair cases combine straightforwardly. Case ABC is $\text{CCD}_{4,2}^{1D}$. Degree choices are minimal while ensuring correctness. Notably, A has low effect relative to B or C. The total speedup is almost 4x.

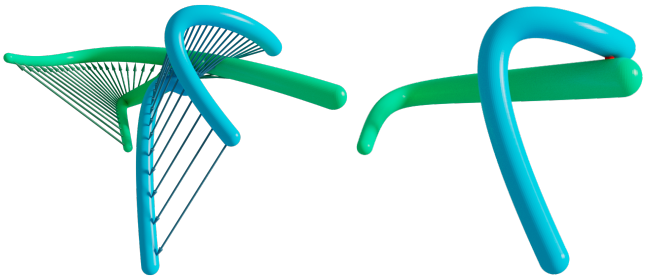


Figure 4: Example of collision detection between two TCCs. On the left, the TCC is visualized at $t = 0$ (tail of arrows) and $t = 1$ (head of arrows), showing the deformation due to the spline centerline parameters varying linearly in time. On the right, the first collision time is shown, with the earliest collision marked by a red point.

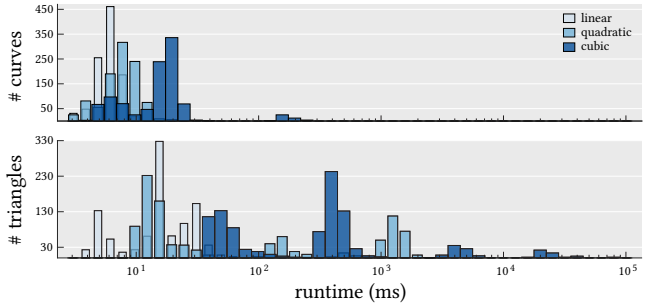
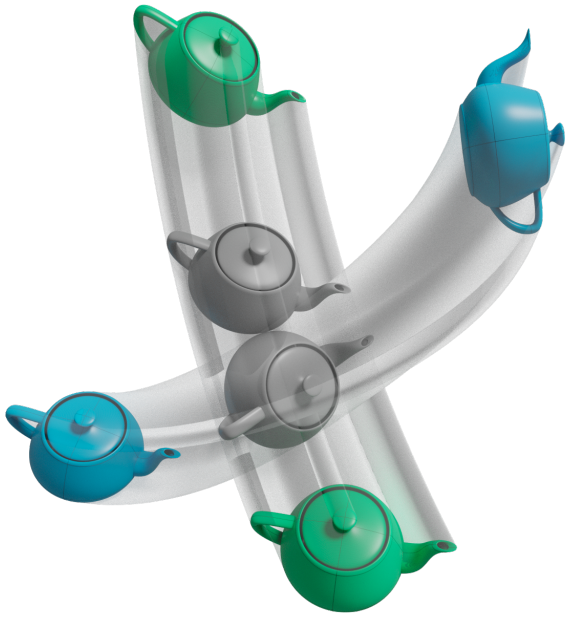
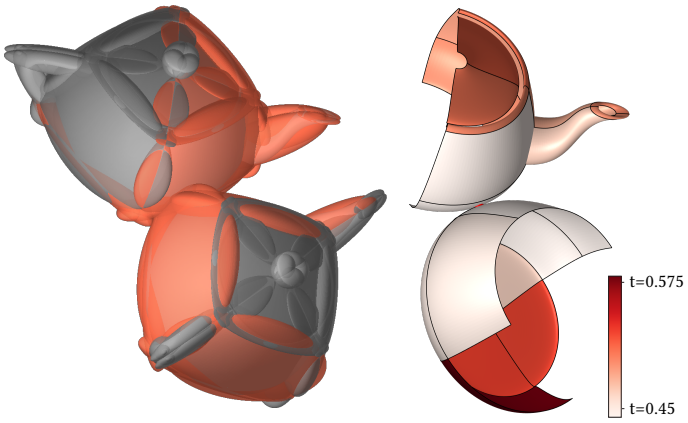


Figure 5: Log scale runtime histograms on batch collision detection problems with culling. Runtimes for linear geometries are unsurprisingly lower than for higher degree geometries. The effect of culling is evident from the isolated runtime modes. These are especially clear in the cubic triangle case.





Geometry	N	C	% Exact	Time (ms)	Size
ellipsoid	32^2	76	0	750	$58^2 + 3 \cdot 10^2$
bicubic patch	76	52	77	$1.6 \cdot 10^5$	$158^2 + 5 \cdot 56^2$

Figure 6: Two bicubic Bézier teapots are swept rigidly along a DQLERP trajectory. Their earliest collision state is shown mid-sweep. Bounding ellipsoids are computed for each bicubic patch and are used to perform a culling step; the bounding ellipsoids for which we find collisions are shown in red. In the last view, we show the patches that were not culled by the ellipsoid CCD problem, each colored based on the time they first collide with any patch in the other teapot. Below, numerical details accompanying the figure are shown. Out of (N) total CCD problems per geometry, we find (C) collisions. The Ellipsoid CCD problem used for culling is much smaller and faster than bicubic patch CCD on DQLERP. Bicubic patch CCD reveals 52 collisions, though only 40 of those obtain exact recovery with $\epsilon_\lambda = 10^{-4}$. Similarly to [Marschner et al. 2021], cases without exact recovery are still correct by inspection and in all cases tested the optimal point provides a certificate of collision.