Computational Visual Media
DOI 10.1007/s41095-016-0042-8 Vol. 2, No. 2, June 2016, 119–130

Research Article

Efficient and robust strain limiting and treatment of simultaneous collisions with semidefinite programming

Zhendong Wang[1] (✉), **Tongtong Wang**[1], **Min Tang**[1], **and Ruofeng Tong**[1]

Abstract We present an efficient and robust method which performs well for both strain limiting and treatment of simultaneous collisions. Our method formulates strain constraints and collision constraints as a serial of linear matrix inequalities (LMIs) and linear polynomial inequalities (LPIs), and solves an optimization problem with standard convex semidefinite programming solvers. When performing strain limiting, our method acts on strain tensors to constrain the singular values of the deformation gradient matrix in a specified interval. Our method can be applied to both triangular surface meshes and tetrahedral volume meshes. Compared with prior strain limiting methods, our method converges much faster and guarantees triangle flipping does not occur when applied to a triangular mesh. When performing treatment of simultaneous collisions, our method eliminates all detected collisions during each iteration, leading to higher efficiency and faster convergence than prior collision treatment methods.

Keywords strain limiting; collision response; linear matrix inequality (LMI); semidefinite programming

1 Introduction

1.1 Strain limiting

In the real world, many materials, such as cloth and animal tissues, can only deform to a limited degree. Although compliant to small deformations, they are highly resistant to deformations larger than some threshold. For cloth, its structure of fibers and threads easily accommodates small-to-moderate amounts of stretching, but once the structural slack has been taken up, resistance to further stretching increases dramatically [1]. When developing characteristic dynamics models to simulate such materials, many researchers have taken account of this biphasic property as a key to developing wrinkle patterns observed in many fabrics. Similarly, animal tissues, such as skin or relaxed muscles, are also compliant to small strains but very tough and resistant to larger ones.

Unfortunately, most simulation methods perform poorly for materials with highly non-compliant constitutive regimes. Standard finite-element methods, including mass-spring systems, model strong resistance using large material coefficients, leading to integration problems over time [1]. Explicit integration methods require very small time steps to stay stable and avoid divergence. Implicit methods can maintain stability using relatively large time steps, but may converge slowly, and suffer from high residuals or excessive damping. To prohibit excessive extensibility in simulation, most dynamic models use projection to enforce a hard limit on large strains, i.e., *strain limiting*.

Many strain limiting methods have been proposed. Anisotropic strain limiting methods [2] in cloth simulation limit strains of edges. Isotropic methods [1, 3] act on the strain tensors and limit the singular values of the deformation gradient matrices of triangles. Our method also performs isotropic strain limiting for both triangular surface meshes and tetrahedral volume meshes. Compared to prior methods, our method has faster convergence and produces better strain limited triangular meshes without any flipped triangles.

1 College of Computer Science and Technology, Zhejiang University, China. E-mail: Z. Wang, wangzhendong@zju.edu.cn (✉); T. Wang, wtt923@zju.edu.cn; M. Tang, tang_m@zju.edu.cn; R. Tong, trf@zju.edu.cn.

Manuscript received: 2015-12-02; accepted: 2016-01-14

1.2 Treatment of collisons

Collision handling is another vital research area in computer animation. Deformable bodies naturally bring about large numbers of collisions varying in strength from resting contact to high speed impact. Collisions between soft and thin sheets, such as cloth and paper, are especially difficult to handle. There are two phases in collision handling, collision detection and collision response. Collision detection is an important component of physically based simulation systems targeting cloth and hair, and finite-element simulation. Even a single undetected collision can lead to simulation failure. Continuous collision detection (CCD) algorithms are widely used in cloth simulation to detect collisions between cloths, cloths and rigid bodies, and self-collisions of the same cloth. Robust collision response is also vital for cloth and shell simulation. Not only must collisions be prevented, but the response must be physically plausible. Unsuitable treatment of collisions may lead to divergence in collision handling and simulation failure. While the handling of individual collisions is well understood, simultaneous collisions can halt existing methods. When facing a cluster of interacting simultaneous collisions, a *rigid impact zone* (RIZ) approach as proposed by Bridson et al. [4] may be adopted to prevent collisions, but this also eliminates all relative tangential velocity in a physically implausible way. Another approach based on *inelastic projection impact zone* (IIZ), proposed by Harmon et al. [5], formulates collisions as linear constraints and then solves a nearest projection optimization problem. Our method is similar to IIZ, but we impose an additional determinant inequality constraint on each collision which ensures we can eliminate every detected collision in only one iteration.

1.3 Semidefinite programming

The notation $\boldsymbol{A} \succeq 0$ implies that $\boldsymbol{A}$ is a symmetric, positive semidefinite (PSD) matrix. Such an expression is a linear matrix inequality (LMI) [6]. In the same manner, $\boldsymbol{A} \succeq \boldsymbol{B}$ implies that $\boldsymbol{A} - \boldsymbol{B}$ is PSD and thus for a scalar $c \in \mathbb{R}$, the equation $\boldsymbol{S} \succeq c\boldsymbol{I}$ implies that the eigenvalues of $\boldsymbol{S}$ are greater than or equal to c.

A semidefinite program (SDP) is a convex optimization problem formulated with LMI constraints and a linear objective. Semidefinite programming unifies several standard problems. It is easy to formulate any linear program, convex quadratic program, and second-order cone program as an SDP. SDP solvers are still not as mature as more classical optimization methods, and have higher time complexity. However, they are already efficient enough for many applications in computer graphics. Semidefinite programs are as easy to solve as linear programs. Most interior-point methods have been generalized to semidefinite programs.

1.4 Main results

In this paper, we present an efficient and robust method to perform both strain limiting and treatment of simultaneous collisions.

- We perform strain limiting of triangular meshes in the fashion of strain limiting of tetrahedral meshes. The benefit is that it can prevent triangle flipping.
- We impose an additional determinant constraint on collision response, which can ensure the detected collisions to be eliminated by only one iteration.
- We formulate strain constraints and collision constraints as a serial of LMIs and LPIs, then solve a projection optimization problem with semidefinite programming.

1.5 Organization

The rest of the paper is organized as follows. We survey prior work on strain limiting and collision handling in Section 2. We present a position based projection optimization problem with strain and collision constraints in Section 3. We present our semidefinite programming solution to our optimization problem in Section 4; implementation is considered in Section 5. We highlight the results and performance of our method in Section 6.

2 Related work

Provot [7] introduced strain limiting as a technique for stably modeling stiff springs by imposing constraints on the maximum and minimum allowed strain of each link. Subsequently, many extensions of this technique have been developed. Bridson et al. [4] applied momentum-conserving impulses to the velocities to ensure that all springs are deformed by a maximum of 10% from their rest lengths at the end

of each time step. Goldenthal et al. [8] proposed an efficient constrained Lagrangian method for modeling inextensible spring networks. For triangle meshes, English and Bridson [9] used nonconforming elements to allow more degrees of freedom of a strain limited triangle to model inextensible cloth. Thomaszewski et al. [10] presented a continuum-based technique that independently constrains the three components of the strain tensor of a triangle. A technique for isotropic strain limiting was proposed by Wang et al. [1], who also introduced a multi-resolution approach for enforcing these constraints. Narain et al. [3] posed strain limiting as a nonlinear optimization problem and used an augmented Lagrangian method to solve it.

Collision handling plays a significant role in physically based simulation. Continuous collision detection (CCD) methods are widely used to detect collisions and intersections in cloth, hair, and thin sheet simulations. Brochu et al. [11] proposed a volume-based geometric predictor, and Tang et al. [12] proposed a Bernstein sign classification (BSC) based predictor, both of which can provide exact collision results by taking advantage of exact geometric arithmetic. Wang [13] introduced error analysis into a traditional cubic solver for CCD to achieve conservative but acceptable collision results. Detected collisions also need to be handled properly. Bridson et al. [4] applied repulsion impulses to collision elements to remove collisions. To handle clusters of interacting simultaneous collisions that repulsion impulses are not able to deal with, Provot [14] and Bridson et al. [4] used a rigid impact zone (RIZ) approach to prevent collisions. Huh et al. [15] divided the impact zone into clusters to partially alleviate the rigidification. Tsiknis [16] considered shearing modes of the impact zone. Harmon et al. [5] proposed a new inelastic projection impact zone (IIZ) method to better deal with simultaneous collisions. Tang et al. [17] presented a method to compute continuous penalty forces to determine collision responses between rigid and deformable models bounded by triangle meshes.

3 Position based projection

Many approaches to the simulation of dynamic systems in computer graphics are force based methods. In physically based simulation, strain limiting and collision response are used as remedies when excessive deformations or collisions appear after numerical time integration. Force based methods [4, 5, 18] act on velocities and then use the updated velocities to find final positions meeting with various kinds of constraints, such as strain constraints, collision constraints, and position constraints (e.g., fixed points). It usually needs many iterations to determine final good-quality positions, and requires relatively small time steps to keep the simulation system stable. In contrast, Müller et al. [19] proposed position based dynamics (PBD), which acts directly on positions to get a well constrained simulation configuration. Many constraints, such as strain and collision constraints, are very easy to handle by projecting points to valid locations in PBD. It is also very stable and allows simulations to take relatively large time steps.

Strain limiting and collision response can both be viewed as finding the closest projections to meshes which are correctly strain limited and collision-free. Given a mesh with m vertices, we can get its candidate positions $[\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m]^{\mathrm{T}} = \boldsymbol{q} \in \mathbb{R}^{3m}$ after numerical time integration. Let $\boldsymbol{q}' \in \mathbb{R}^{3m}$ be the well constrained positions of the mesh. The objective of the position based projection optimization problem is defined as following:

$$\min_{\boldsymbol{q}'} \quad ||\boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{q}' - \boldsymbol{q})||_{\mathrm{F}}^2 \tag{1}$$

where $\boldsymbol{M}$ is the mass matrix and $||.||_{\mathrm{F}}$ denotes the Frobenius norm. Strain constraints and collision constraints are imposed respectively when performing strain limiting and collision response. We adopt semidefinite programming to solve the optimization problem stably and efficiently, as presented in Section 4.

3.1 Strain constraints

A well strain-limited surface mesh is one in which the strains of all faces lie within the user-specified strain limits $[s_{\min}, s_{\max}]$. We use $[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]^{\mathrm{T}} = \boldsymbol{u} \in \mathbb{R}^{2m}$ to denote the material coordinates of a surface mesh in material space, with $\boldsymbol{u}_i \in \mathbb{R}^2$. The deformation gradient of a triangle face is $\boldsymbol{F} = \boldsymbol{D}_{\boldsymbol{q}}\boldsymbol{D}_{\boldsymbol{u}}^{-1}$, where $\boldsymbol{D}_{\boldsymbol{q}} = [\boldsymbol{q}_j - \boldsymbol{q}_i, \boldsymbol{q}_k - \boldsymbol{q}_i]$ is a 3×2 matrix and $\boldsymbol{D}_{\boldsymbol{u}} = [\boldsymbol{u}_j - \boldsymbol{u}_i, \boldsymbol{u}_k - \boldsymbol{u}_i]$ is a 2×2 matrix [1]. We diagonalize $\boldsymbol{F}$ into $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}$ using singular value decomposition (SVD); $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal

matrices and $\boldsymbol{S}$ is a 3×2 matrix with nonnegative values on the diagonal, which are the two principle strains (s_1, s_2) of the triangle face. So the strain constraints for the triangle face are

$$s_{\min} \leqslant s(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k) = (s_1, s_2) \leqslant s_{\max} \quad (2)$$

i.e., $s_{\min} \leqslant s_1, s_2 \leqslant s_{\max}$.

Strain limiting for a volume mesh is performed in a similar way. The deformation gradient of a tetrahedron $[\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l]$ is also written as $\boldsymbol{F}$. Unlike the surface mesh case, now $\boldsymbol{D_q} = [\boldsymbol{q}_j - \boldsymbol{q}_i, \boldsymbol{q}_k - \boldsymbol{q}_i, \boldsymbol{q}_l - \boldsymbol{q}_i]$ and $\boldsymbol{D_u} = [\boldsymbol{u}_j - \boldsymbol{u}_i, \boldsymbol{u}_k - \boldsymbol{u}_i, \boldsymbol{u}_l - \boldsymbol{u}_i]$ are 3×3 matrices because $\boldsymbol{u}_i \in \mathbb{R}^3$ for a volume mesh. Finding the SVD of $\boldsymbol{F}$ gives three nonzero singular values (s_1, s_2, s_3), so the strain constraint for a tetrahedron is

$$s_{\min} \leqslant s(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l) = (s_1, s_2, s_3) \leqslant s_{\max} \quad (3)$$

To unify strain limiting for surface meshes and volume meshes, we extend the 2D material coordinates of surface meshes to 3D by simply setting the third value to zero, i.e., $\boldsymbol{u}_i = [u_1, u_2, 0]^{\mathrm{T}}$. In addition, we introduce an auxiliary vertex $\boldsymbol{p}_l$ for each triangle $[\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k]$ of the surface mesh, as shown in Fig. 1(a). Its material coordinate is $\boldsymbol{u}_{\boldsymbol{p}_l} = \boldsymbol{u}_i + [0, 0, 1]^{\mathrm{T}}$, and its initial position in world space is $\boldsymbol{p}_l = \boldsymbol{q}_i + \boldsymbol{n}$ where $\boldsymbol{n}$ is the unit normal vector of the triangle. This transforms a triangular surface into a tetrahedral volume mesh, as shown in Fig. 1(b). This allows Eq. (2) to be updated to

$$s_{\min} \leqslant s(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{p}_l) = (s_1, s_2, s_3) \leqslant s_{\max} \quad (4)$$

Moreover, we add a matrix determinant inequality constraint as below to make sure that strain limited triangles do not flip; a comparison with the previous method without this constraint is shown in Fig. 2.

$$d(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{p}_l) = \det([\boldsymbol{q}_j - \boldsymbol{q}_i, \boldsymbol{q}_k - \boldsymbol{q}_i, \boldsymbol{p}_l - \boldsymbol{q}_i]) > 0 \quad (5)$$

This indicates that the matrix is orientation preserving and the volume of the tetrahedron

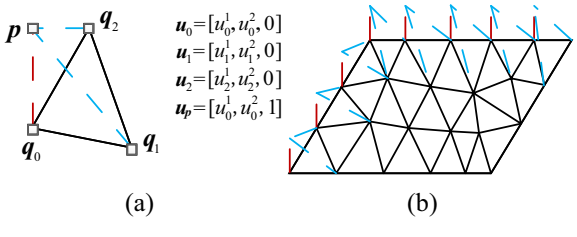


Fig. 1 Transforming a triangular mesh into a tetrahedral mesh. (a) An auxiliary vertex $\boldsymbol{p}$ is added so that the vector $\boldsymbol{q}_0\boldsymbol{p}$ is the unit normal vector of the triangle in both world space and material space. (b) After adding an auxiliary vertex for each triangle, a triangular mesh is transformed into a tetrahedral mesh. Only some of the auxiliary vertices are shown in (b).

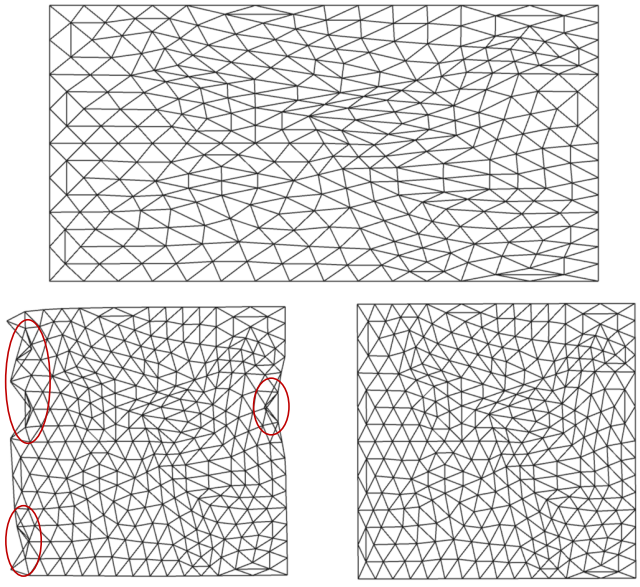


Fig. 2 Strain limiting for a triangular surface mesh. The top mesh is stretched lengthwise by a factor of 1/2 from the original undeformed mesh. The strain of the stretched mesh is limited to lie within [0.99, 1.01]. The bottom-left mesh is produced by Narain et al.'s strain limiting method [3]; some triangles have flipped as highlighted in red. The bottom-right mesh is produced by our method; the strain is well limited, without flipping triangle.

constructed by a triangle and its auxiliary vertex is always positive. The strain constraints for multiple triangles are formulated as follows:

$$s_{\min} \leqslant \boldsymbol{s}(\boldsymbol{q}, \boldsymbol{p}) \leqslant s_{\max} \quad (6a)$$

$$\boldsymbol{d}(\boldsymbol{q}, \boldsymbol{p}) > 0 \quad (6b)$$

If there are w triangles, then $[\boldsymbol{p}_1, ..., \boldsymbol{p}_w]^{\mathrm{T}} = \boldsymbol{p} \in \mathbb{R}^{3w}$, $\boldsymbol{s} \in \mathbb{R}^{3w}$ and $\boldsymbol{d} \in \mathbb{R}^w$.

3.2 Collision constraints

When performing treatment of collisions, two elementary kinds of primitive, vertex-face (VF) pairs and edge-edge (EE) pairs, need to be tackled properly. Each pair consists of four vertices, $[\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l]$. For a VF pair, $\boldsymbol{q}_i$ is a vertex and $[\boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l]$ represents a triangle face. For an EE pair, $[\boldsymbol{q}_i, \boldsymbol{q}_j]$ and $[\boldsymbol{q}_k, \boldsymbol{q}_l]$ represent the two edges. Two distance constraint functions for a VF and an EE pair are defined as respectively in Ref. [5], as follows:

$$\begin{aligned} c_{\mathrm{VF}}(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l) &= \boldsymbol{n} \cdot [\alpha_0 \boldsymbol{q}_i - (\alpha_1 \boldsymbol{q}_j + \alpha_2 \boldsymbol{q}_k + \alpha_3 \boldsymbol{q}_l)] \\ c_{\mathrm{EE}}(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l) &= \boldsymbol{n} \cdot [(\alpha_0 \boldsymbol{q}_i + \alpha_1 \boldsymbol{q}_j) - (\alpha_2 \boldsymbol{q}_k + \alpha_3 \boldsymbol{q}_l)] \end{aligned} \quad (7)$$

where $\boldsymbol{n}$ is a normal vector and $\alpha_0, \alpha_1, \alpha_2, \alpha_3 \in [0, 1]$ are parameters. For a VF pair, $\boldsymbol{n}$ is the normal of the triangle face, $\alpha_0 = 1$, $\alpha_1 + \alpha_2 + \alpha_3 = 1$. For an EE pair, $\boldsymbol{n}$ is the cross product of the two edges, $\alpha_0 + \alpha_1 = 1, \alpha_2 + \alpha_3 = 1$.

A pair is collision-free provided that $c > 0$. At an intermediate time in a time step, a pair is colliding if $c \leqslant 0$, as shown in Fig. 3; the projection of vector $\boldsymbol{q}_c\boldsymbol{q}_i$ onto the normal vector $\boldsymbol{n}$ is negative. So the collision constraint for a pair is

$$c(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l) > 0 \tag{8}$$

Because the normal $\boldsymbol{n}$ dependends on the positions of the four vertices, the collision constraints are nonlinear. Harmon et al. [5] just sets $\boldsymbol{n}$ to be the normal at the time of collision, so the constraint function c becomes linear. This corresponds to removing collisions by just modifying the positions along the normal vector, which essentially conforms to the laws of physics. Although the linear constraint in Eq. (8) is met, it cannot guarantee the elimination of collisions at the end of a time step because the vertex may still be on the negative side of the triangle face for a VF pair, like in the case shown in Fig. 3(b).

To simplify $c(\boldsymbol{q})$, we also make the same choice as Harmon et al. [5]. In the meantime, we impose a similar constraint to Eq. (5) on collision:

$$d(\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_k, \boldsymbol{q}_l) = \det([\boldsymbol{q}_j - \boldsymbol{q}_i, \boldsymbol{q}_k - \boldsymbol{q}_i, \boldsymbol{q}_l - \boldsymbol{q}_i]) > 0 \tag{9}$$

which indicates the volume of the tetrahedron consisting of the four vertices of a collision pair is positive, i.e., a vertex is always on the positive side of its opposite triangle in the tetrahedron. Combining Inequality (9) with Inequality (8) simplifies the processing of collisions and ensures that collisions are eliminated. Figure 3(c) shows a result that imposes constraints with both Inequalities (8) and (9), ensuring that the vertex is on the positive side of the triangle face. Multiple collisions can be constrained simultaneously.

$$\boldsymbol{c}(\boldsymbol{q}) > 0 \tag{10a}$$

$$\boldsymbol{d}(\boldsymbol{q}) > 0 \tag{10b}$$

If there are k collisions, then $\boldsymbol{c} \in \mathbb{R}^k$, $\boldsymbol{d} \in \mathbb{R}^k$.

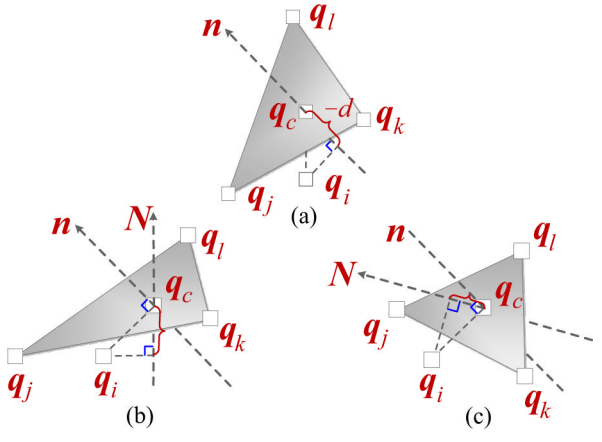


Fig. 3 Collision response for a VF pair. A VF collision pair consists of vertex $\boldsymbol{q}_i$ and triangle $\Delta\boldsymbol{q}_j\boldsymbol{q}_k\boldsymbol{q}_l$. $\boldsymbol{n}$ is the normal vector at collision time and $\boldsymbol{q}_c = \alpha_1\boldsymbol{q}_j + \alpha_2\boldsymbol{q}_k + \alpha_3\boldsymbol{q}_l$ is the collision point, where $\alpha_1, \alpha_2, \alpha_3$ are barycentric coordinates. $\boldsymbol{N}$ is the normal vector of the triangle after the collision response.

3.3 Positional constraints

Positional constraints, such as fixed-points and gluing constraints, are very common in cloth simulation.

3.3.1 Fixed points

A fixed-point constraint for vertex i can be formulated as

$$f(\boldsymbol{q}_i) = ||\boldsymbol{q}_i - \boldsymbol{x}_i||_1 = 0 \tag{11}$$

in which $\boldsymbol{x}_i$ is a fixed point in world space. The fixed-point constraint function can be reformulated to enforce multiple fixed-point constraints using:

$$f(\boldsymbol{q}) = ||\boldsymbol{P}\boldsymbol{q} - \boldsymbol{x}||_1 \tag{12}$$

$[\boldsymbol{x}_1, ..., \boldsymbol{x}_m]^{\mathrm{T}} = \boldsymbol{x} \in \mathbb{R}^{3m}$ is a column vector and $\boldsymbol{P}$ is an $m \times m$ diagonal block matrix, where each block is a 3×3 matrix. If vertex i is constrained, the ith entry $\boldsymbol{x}^i$ is a user-defined vector in world space and the ith entry on the diagonal of $\boldsymbol{P}$ is an identity matrix, i.e., $\boldsymbol{P}_{ii} = \boldsymbol{I}$. The other entries of $\boldsymbol{x}$ and $\boldsymbol{P}$ are zero.

3.3.2 Glue

A gluing constraint for two vertices i and j can be formulated as

$$d_{\min} \leqslant g(\boldsymbol{q}_i, \boldsymbol{q}_j) = \boldsymbol{n}^{\mathrm{T}} \cdot (\boldsymbol{q}_i - \boldsymbol{q}_j) \leqslant d_{\max} \tag{13}$$

where $\boldsymbol{n} \in \mathbb{R}^3$ is a unit normal and $[d_{\min}, d_{\max}]$ is a distance interval. The glue constraint function can also be reformulated to enforce multiple glue constraints using:

$$\boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{N}\boldsymbol{q} \tag{14}$$

If there are k glue constraints, $\boldsymbol{G} \in \mathbb{R}^k$ and $\boldsymbol{N} \in \mathbb{R}^{k\times3m}$, which is a sparse matrix. Each row of $\boldsymbol{N}$ is in the format $[\boldsymbol{n}_1, ..., \boldsymbol{n}_m]^{\mathrm{T}} = \boldsymbol{N}_{\mathrm{row}}^{\mathrm{T}} \in \mathbb{R}^{3m}$. If vertices i and j are glued together, then $\boldsymbol{n}_j = -\boldsymbol{n}_i$ and the other entries are zero.

4 Solution by semidefinite programming

Strain constraints (i.e., Inequalities (4) and (5)) and collision constraints (i.e., Inequality (9)) are nonlinear and non-convex, which makes the projection optimization problem complicated and difficult to

solve. Narain et al. [3] adapted an augmented Lagrangian method to solve the problem with strain constraints. However, it converges slowly and is not robust in practice, as shown in Figs. 2 and 4. Instead, to solve our problem, we adopt the method proposed by Kovalsky et al. [20], which can control the singular values of a square matrix to lie within a positive interval $[\gamma, \Gamma]$ by use of semidefinite programming (SDP).

A *meta-problem* is defined as follows to control the singular values of an arbitrary square matrix.

$$\min_{\boldsymbol{A}\in\mathbb{R}^{n\times n}} f(\boldsymbol{A}, s_{\min}(\boldsymbol{A}), s_{\max}(\boldsymbol{A})) \tag{15a}$$

$$\text{such that} \quad s_{\max}(\boldsymbol{A}) \leqslant \Gamma \tag{15b}$$

$$s_{\min}(\boldsymbol{A}) \geqslant \gamma, \quad \det(\boldsymbol{A}) \geqslant 0 \tag{15c}$$

where $\boldsymbol{A}$ is an $n\times n$ matrix, $s_{\min}(\boldsymbol{A})$ and $s_{\max}(\boldsymbol{A})$ are the minimum and maximum singular values respectively, and $f(\boldsymbol{A}, s_{\min}(\boldsymbol{A}), s_{\max}(\boldsymbol{A}))$ is a convex objective function. Obviously, the feasible set of the meta-problem is non-linear and non-convex. It is difficult to solve it using traditional optimization methods. Recently, Kovalsky et al. [20] proposed a simple and efficient method to solve the meta-problem, which reformulates the non-linear and non-convex constraints in Inequalties (15b) and (15c) as two linear matrix inequalities respectively.

Inequality (15b) can be replaced equivalently by the following LMI:

$$\begin{pmatrix} \Gamma\boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & \Gamma\boldsymbol{I} \end{pmatrix} \succeq 0 \tag{16}$$

Furthermore, a family of maximal convex subsets is found to cover the entire set defined by Inequality (15c). Each maximal convex subset is defined by an LMI of the form:

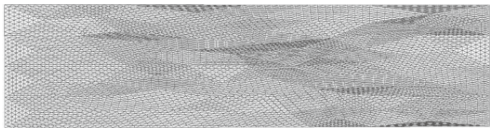
$$\frac{\boldsymbol{A}+\boldsymbol{A}^{\mathrm{T}}}{2} \succeq \gamma\boldsymbol{I} \tag{17}$$

To find a global solution, the procedure rotates the current maximal convex subset to the next iteratively.

Positional constraints such as fixed-point and glue constraints are easy to handle because they are linear. We just detail how to deal with strain constraints and collision constraints. The problem of strain limiting for a single triangle or tetrahedron, or collision for a single pair, is very similar to the *meta-problem*. Strain and collision constraints can be reformulated as LMIs and LPIs, allowing us to take advantage of standard convex SDP solvers to solve strain limiting and collision problems in physically based simulation.

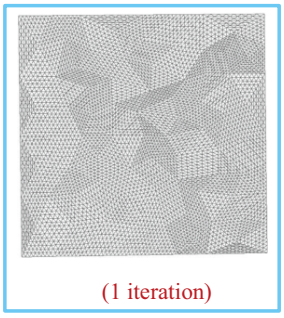
4.1 Strain limiting

Strain limiting for a single triangle can be defined via the following projection optimization problem with LMI constraints:

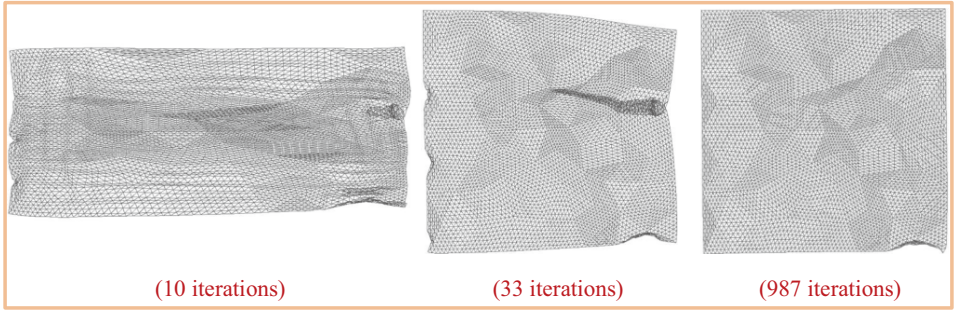


(a)

	Our method	Narain et al.'s method		
Iteration	1	10	33	987
Average time per iteration (s)	93.79	0.775	2.027	1.071
Total time (s)	93.79	7.75	66.88	1057



(b) Our method



(c) Narain et al.'s method

Fig. 4 Comparison of strain limiting methods. The triangular mesh at top right (a) has 9600 triangles; it is stretched twice in length and compressed by half in height compared to the original un-deformed mesh. Mesh (b) was generated by our method. As the picture shows, our method generates the best result. The meshes in (c) were generated by Narain et al.'s strain limiting method [3] using 10, 33, and 987 iterations respectively. The table at top right shows the number of iterations taken by these two methods to converge. It is clear that our method converges faster (in one iteration) than Narain et al.'s method [3]. The disadvantage of our method is that each iteration takes longer to perform.

$$\min_{\boldsymbol{q}'} \quad ||\boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{q}'-\boldsymbol{q})||_{\mathrm{F}}^2 \tag{18a}$$

$$\text{such that} \quad \begin{pmatrix} \Gamma\boldsymbol{I} & \boldsymbol{F} \\ \boldsymbol{F}^{\mathrm{T}} & \Gamma\boldsymbol{I} \end{pmatrix} \succeq 0 \tag{18b}$$

$$\frac{\boldsymbol{F}+\boldsymbol{F}^{\mathrm{T}}}{2} \succeq \gamma\boldsymbol{I} \tag{18c}$$

where $\boldsymbol{F}$ is the deformation gradient matrix of the triangle defined in Section 3.1. After transforming the triangle into a tetrahedron, $\boldsymbol{F}$ is a square matrix. Because $\boldsymbol{F}$ is just a linear transformation of $\boldsymbol{q}$, we can easily apply the method to our strain limiting problem. Having shown how strain limiting for a single triangle or tetrahedron is done, it is easy to extend Eq. (18) to the case for multiple triangles and tetrahedra.

4.2 Collision response

For a single pair involved in a collision, the problem can be defined as a projection optimization problem with both LMI and LPI constraints:

$$\min_{\boldsymbol{q}'} \quad ||\boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{q}'-\boldsymbol{q})||_{\mathrm{F}}^2 \tag{19a}$$

$$\text{such that} \quad c(\boldsymbol{q}') > 0 \tag{19b}$$

$$\frac{\boldsymbol{q}'\Delta+(\boldsymbol{q}'\Delta)^{\mathrm{T}}}{2} \succeq \gamma\boldsymbol{I} \tag{19c}$$

where $\boldsymbol{q}=[\boldsymbol{q}_i,\boldsymbol{q}_j,\boldsymbol{q}_k,\boldsymbol{q}_l]$ represents the collision pair, and $\boldsymbol{q}\Delta=[\boldsymbol{q}_j-\boldsymbol{q}_i,\boldsymbol{q}_k-\boldsymbol{q}_i,\boldsymbol{q}_l-\boldsymbol{q}_i]$ is a 3×3 matrix which is a linear transformation of $\boldsymbol{q}$. Inequality (19b) is linear so that we can also apply this method to the collision problem. Furthermore, scenarios with complex simultaneous collisions are also very easy to handle.

5 Implementation

5.1 Local strain limiting

There are two manners in which we can perform strain limiting for a deformable mesh, global and local. In global strain limiting, the candidate positions of all vertices in a mesh are constrained simultaneously. Global strain limiting is very simple, but relatively slow.

In local strain limiting, we detect triangles which violate strain limits. Correlated triangles which share vertices are put into an *SL zone* which represents the regions where strain limiting is needed, just like an *impact zone* which is widely used in collision response methods to deal with complex scenes with simultaneous collisions. We take each SL zone as a unit when dealing with regions which violate strain constraints. For each SL zone, a projection optimization problem is solved to project the region to the nearest location which satisfies the strain constraints. Local strain limiting needs to detect triangles violating strain limits and handle them iteratively until no triangles are detected. Each detected triangle is a smallest SL zone. Correlated SL zones are merged with each other. The extreme case is that the entire mesh is covered by a sinlge SL zone, which is equivalent to global strain limiting.

Handling an SL zone may make correctly strain limited triangles become no longer strain limited, necessitating another iteration. The worst case is that no longer strain limited triangles may appear one by one, causing slow local strain limiting convergence, and taking much time. To accelerate convergence, we extend each SL zone by merging it with its one ring of neighbouring triangles, as shown in Fig. 5. This approach makes SL zones enlarge more quickly, contributing to faster convergence of local strain limiting.

When applying global strain limiting in physically based simulations, the internal energies stored in meshes propagate faster and better than in local strain limiting. The disadvantage is that it takes relatively more time to solve a big optimization problem, especially when meshes are generally already correctly strain limited. In contrast, local strain limiting is very fast when meshes are correctly strain limited. As SL zones expand, it takes more and more iterations to ensure the meshes retain good qualities. Additionally, detecting no longer strain limited triangles in each iteration is expensive. Thus, global strain limiting is more suitable for simulations using large time steps where large deformations

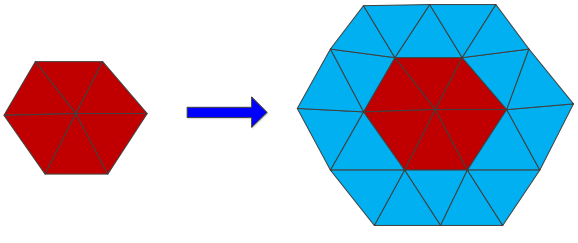


Fig. 5 Local strain limiting. The red region is an SL zone. Before performing strain limiting for this SL zone, we extend it by merging with its one ring neighbour triangles, i.e., the light blue region in the left. Then we perform strain limiting for the extended bigger SL zone. This helps local strain limiting converge faster.

appear frequently, while local strain limiting is more suitable to simulations using small time steps.

5.2 SDP optimization

In our method, we have to solve the optimization problem defined in Eq. (20), in which the objective is a quadratic function:

$$\min_{\boldsymbol{q}'} \quad ||\boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{q}'-\boldsymbol{q})||_{\mathrm{F}}^2 \qquad \text{subject to LMI and LPI constraints} \tag{20}$$

To solve the problem with an SDP solver, we reformulate the problem as the following equivalent optimization problem with linear objective:

$$\begin{aligned} \min \quad & t && (21a)\\ \text{subject to} \quad & ||\boldsymbol{M}^{\frac{1}{2}}\boldsymbol{z}||_{\mathrm{F}} \leqslant t && (21b)\\ & \boldsymbol{q}'-\boldsymbol{q}=\boldsymbol{z} && (21c)\\ & \text{LMI and LPI constraints} && (21d) \end{aligned}$$

in which $\boldsymbol{z}$ and t are auxiliary variables. Inequality (21b) is a convex conic quadratic constraint and Inequality (21c) is a linear constraint. The new optimization problem is easy to solve with a standard convex SDP solver.

6 Results

We reformulate strain constraints and collision constraints as a series of linear matrix inequalities and linear polynomial inequalities in the projection optimization problem. The transformed problem is easy to solve using standard convex semidefinite programming solvers; we use the one in Mosek [21].

Compared to Narain's strain limiting method [3], our method converges faster and can prevent triangle flipping when performing strain limiting for triangular meshes. Figure 6 shows that our method performs well in strain limiting for tetrahedral meshes. However, Narain et al.'s method [3] takes less time in each strain limiting iteration than our method. Compared with Harmon et al.'s collision response method [5], our method takes fewer iterations and less time to converge, making the collision handling process faster.

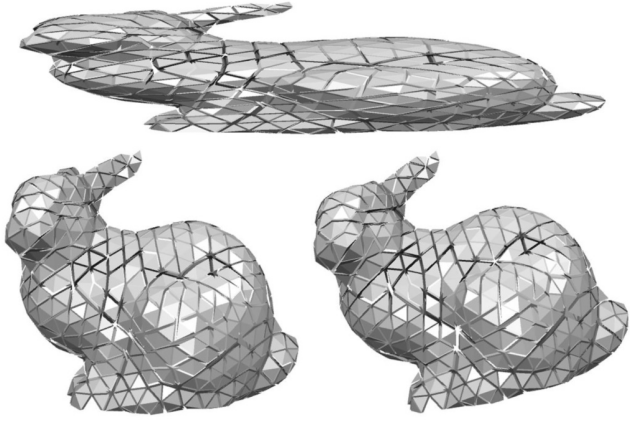


Fig. 6 Strain limiting for a tetrahedral volume mesh. Bottom left: original un-deformed mesh. Top: the mesh is stretched to twice its original length and compressed to half its original height. Bottom right: the mesh produced by our method has well limited strain and is closest to the deformed mesh.

6.1 Performance

We evaluated the performance of our method with various cloth simulation benchmarks, as shown in Figs. 7, 8, and 9, using a standard PC (3.4 GHz Intel i7-4770 CPU, 8 GB RAM, 64-bit Windows 7, NVIDIA GeForce GTX 780 GPU). This includes a C++ implementation of strain limiting and collision response based on Mosek's semidefinite programming solver. Figure 9 illustrates results of our method when performing strain limiting on cloth simulations, as well as a comparison with a prior strain limiting method. Table 1 highlights the performance of our method when computing collision responses on different benchmarks, as shown in

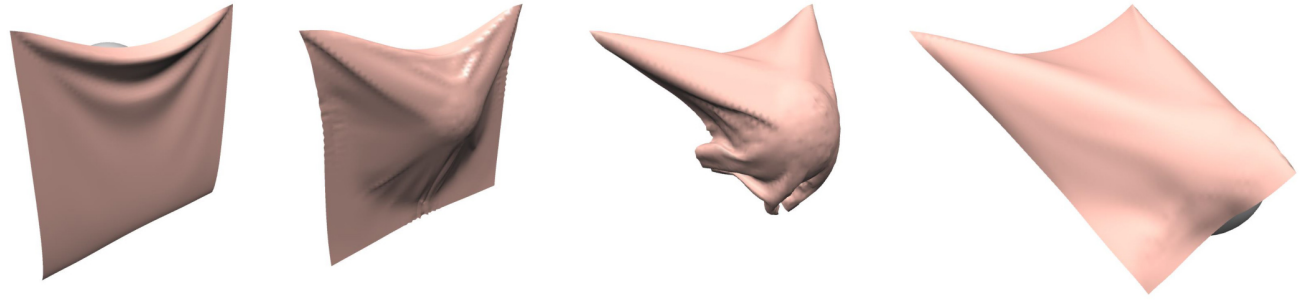
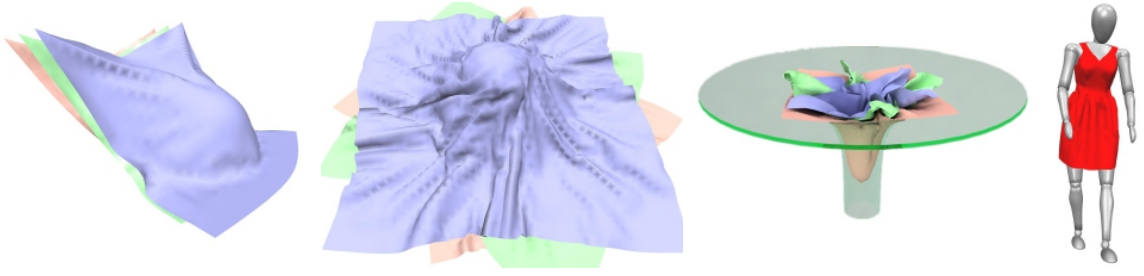


Fig. 7 A moving ball hits a hanging cloth. The cloth mesh has 8.2k triangles; the strains of each triangle are limited to lie in [0.95, 1.05]. This benchmark uses our method to perform strain limiting and collision response.



(a) Hanging cloth (b) Twisting cloth (c) Funnel cloth (d) Human cloth

Fig. 8 Collision response benchmarks: (a) a ball hits three layers of cloth; (b) three layers of cloth fall on a rotating ball and are twisted by it; (c) a falling ball pushes three layers of cloth through a funnel. All of (a), (b), and (c) produce many simultaneous complex collisions which may lead to cloth simulation failure. (d) is a clothed dancing human, in which less complex collisions occur.

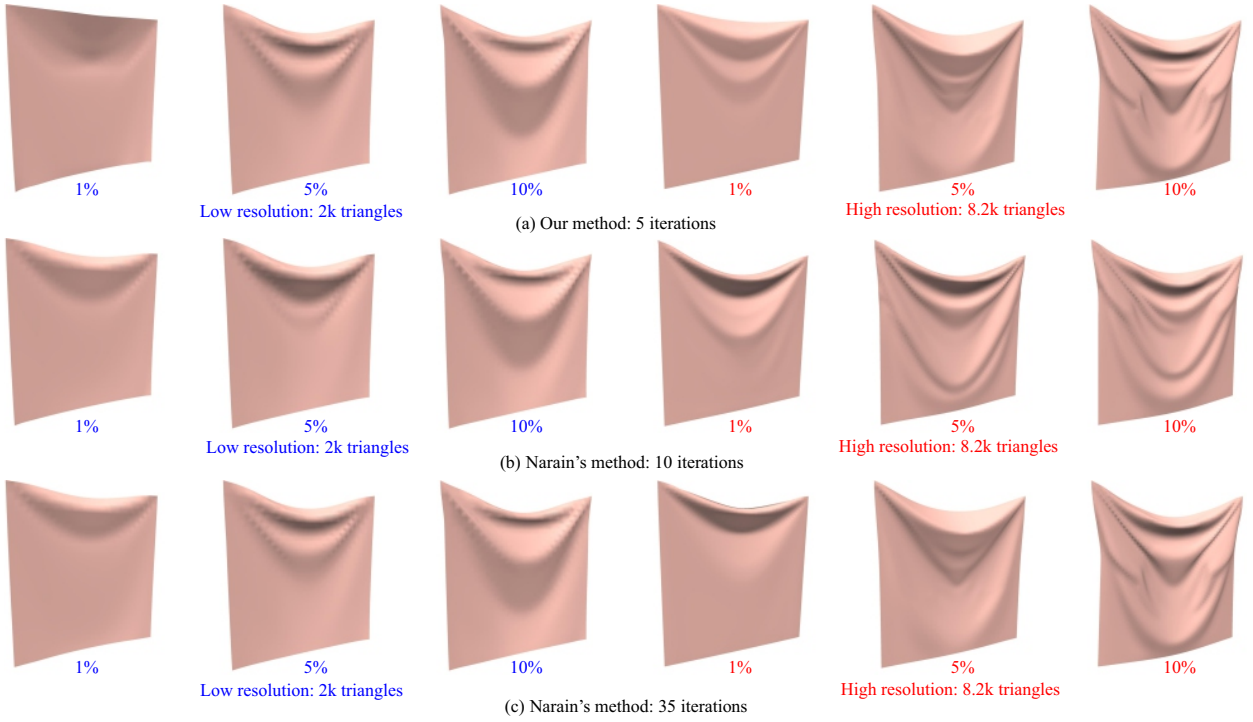


Fig. 9 Strain limiting benchmarks. We use two cloth meshes with different resolutions (2k and 8.2k triangles respectively) to demonstrate the difference between our strain limiting method and Narain et al.'s method during cloth simulation. We limit the deformation of these meshes using different strain limits, [0.9, 1.1] allowing 10% deformation at most compared with the rest mesh, [0.95, 1.05] allowing 5% deformation at most, and [0.99, 1.01] allowing 1% deformation at most, respectively. The cloth exhibits more detailed wrinkles when using a higher resolution mesh. Comparing our method with Narain et al.'s method using the same meshes and the same strain limits, it is clear that meshes generated by our method are better strain limited. In contrast, meshes generated by Narain et al.'s method are more loose only after 10 iterations. After 35 iterationes, Narain et al.'s meshes in the third row become tighter and closer to our meshes in the first row.

Table 1 Comparison of collision response methods. Note the advantages of our collision response method over Harmon et al.'s inelastic projection impact zone approach [5]. Using the same collision detection method, our method takes fewer iterations in each time step to handle simultaneous collisions. Furthermore, our method takes less time in each iteration to deal with multiple simultaneous collisions. Both contribute to a faster collision handling process (including collision detection and collision response)

Benchmarks	Average # of iterations in collision response		Average time of iteration in collision response (s)		Average time of collision handling (s)	
Hanging cloth	4.93	5.53	0.47	0.62	5	6.45
Twisting cloth	6.65	11.63	0.65	0.97	4.23	8.65
Funnel cloth	9.89	12.39	1.61	9.96	20.82	36.42
Human cloth	0.98	1.01	0.0526	0.0014	0.65	0.57

Our collision response method Inelastic projection impact zone

Fig. 8.

In detail we compare the performance of our method with the following methods and implementation:

- Narain et al.'s strain limiting method [3]. Like our method, it also constrains the singular values of the deformation gradient of triangles and solves a projection optimization problem. It takes advantage of an augmented Lagrangian method to transform the constrained optimization problem to an unconstrained problem, and adopts the non-linear conjugate gradient method from the ALGLIB numerical analysis library[1] to obtain its solutions. Narain et al.'s strain limiting method [3] is implmented in ARCSim[2], an open-source simulator. Our method instead performs strain limiting for triangles based on tetrahedra, to prevent triangle flipping. Additionally, our method transforms non-linear strain constraints into linear matrix inequality constraints. We take advantage of Mosek's semidefinite programming solver to solve the optimization problem.
- Harmon et al.'s collision response method [5] using inelastic projection. Inelastic projection is actually a velocity filter because it acts on velocities of meshes. It solves a projection optimization problem with linear equation constraints. In our method, we replace the linear equations by linear inequalities and add additional determinant inequality constraints. This ensures that detected collisions are eliminated after only one iteration, as shown in Fig. 3.

6.2 Analysis

Strain limiting and treatment of collisions are two important processes in physically based simulation, particularly cloth and hair simulations. We have presented an efficient and robust method which can deal with both of them well. There are several advantages of our method. When performing strain limiting, we transform a triangle into a tetrahedron; our method applies to both triangular surface meshes and tetrahedral volume meshes. Additionally, our method can ensure the volume of a tetrahedron to be positive preventing triangle flipping during strain limiting. Compared with prior strain limiting methods, our method converges faster, although our method takes more time in each iteration when performing global strain limiting. Strain limiting for many triangles produces many low-dimensional LMI constraints, many more than the number of variables. Thus, standard SDP solvers may be non-optimal and need more time to find a optimal solution. When handling simultaneous collisions, our method eliminates all detected collisions during every iteration, which contributes to higher efficiency and faster convergence than prior collision handling methods.

1 Sergey Bochkanov and Vladimir Bystritsky, http://www.alglib.net/

2 http://graphics.berkeley.edu/resources/ARCSim/arcsim-0.2.1.tar.gz

7 Limitations, conclusions, and future work

We have presented an efficient and robust method which performs well both for strain limiting and handling simultaneous collisions. In our method, strain constraints and collision constraints are reformulated as a seriesl of linear polynomial inequalities (LPIs) and linear matrix inequalities (LMIs). Our method solves a projection optimization problem with Mosek's standard semidefinite programming solver. We have tested our method on some cloth simulation benchmarks and highlighted its benefits compared to prior strain limiting methods and collision response methods.

Our method has a few *limitations*. When performing strain limiting for a high-resolution mesh in global fashion, our method takes more time than Narain et al.'s method [3]. When combining strain constraints with collisions constraints, our method may be unstable when many collisions occur simultaneously. In the future, it is very possible and indeed essential to optimize our method to make it faster when performing strain limiting for a high-resolution mesh. A more efficient SDP solver may also help to solve the global strain limiting problem faster.

Acknowledgements

This research is supported in part by the National High-tech R&D Program of China (No. 2013AA013903), National Natural Science Foundation of China (No. 61572423), Zhejiang Provincial NSFC (No. LZ16F020003), the National Key Technology R&D Program of China (No. 2012BAD35B01), the Doctoral

Fund of Ministry of Education of China (No. 20130101110133). Ruofeng Tong is partly supported by National Natural Science Foundation of China (No. 61572424).

References

[1] Wang, H.; O'Brien, J.; Ramamoorthi, R. Multiresolution isotropic strain limiting. *ACM Transactions on Graphics* Vol. 29, No. 6, Article No. 156, 2010.

[2] Ma, G.; Ye, J.; Li, J.; Zhang, X. Anisotropic strain limiting for quadrilateral and triangular cloth meshes. *Computer Graphics Forum* Vol. 35, No. 1, 89–99, 2016.

[3] Narain, R.; Samii, A.; O'Brien, J. F. Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 152, 2012.

[4] Bridson, R.; Fedkiw, R.; Anderson, J. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics* Vol. 21, No. 3, 594–603, 2002.

[5] Harmon, D.; Vouga, E.; Tamstorf, R.; Grinspun, E. Robust treatment of simultaneous collisions. *ACM Transactions on Graphics* Vol. 27, No. 3, Article No. 23, 2008.

[6] Boyd, S.; Vandenberghe, L. *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[7] Provot, X. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In: *Graphics Interface*. Canadian Information Processing Society, 147–154, 1995.

[8] Goldenthal, R.; Harmon, D.; Fattal, R.; Bercovier, M.; Grinspun, E. Efficient simulation of inextensible cloth. *ACM Transactions on Graphics* Vol. 26, No. 3, Article No. 49, 2007.

[9] English, E.; Bridson, R. Animating developable surfaces using nonconforming elements. *ACM Transactions on Graphics* Vol. 27, No. 3, Article No. 66, 2008.

[10] Thomaszewski, B.; Pabst, S.; Straßer, W. Continuum-based strain limiting. *Computer Graphics Forum* Vol. 28, No. 2, 569–576, 2009.

[11] Brochu, T.; Edwards, E.; Bridson, R. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics* Vol. 31, No. 4, Article No. 96, 2012.

[12] Tang, M.; Tong, R.; Wang, Z.; Manocha, D. Fast and exact continuous collision detection with Bernstein sign classification. *ACM Transactions on Graphics* Vol. 33, No. 6, Article No. 186, 2014.

[13] Wang, H. Defending continuous collision detection against errors. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 122, 2014.

[14] Provot, X. Collision and self-collision handling in cloth model dedicated to design garments. 1997. Available at https://graphics.stanford.edu/courses/cs468-02-winter/Papers/Collisions_vetements.pdf.

[15] Huh, S.; Metaxas, D. N.; Badler, N. I. Collision resolutions in cloth simulation. In: Proceedings of the 14th Conference on Computer Animation, 122–127, 2001.

[16] Tsiknis, K. D. Better cloth through unbiased strain limiting and physics-aware subdivision. Master Thesis. The University of British Columbia, 2006.

[17] Tang, M.; Manocha, D.; Otaduy, M. A.; Tong, R. Continuous penalty forces. *ACM Transactions on Graphics* Vol. 31, No. 4, Article No. 107, 2012.

[18] Bouaziz, S.; Martin, S.; Liu, T.; Kavan, L.; Pauly, M. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 154, 2014.

[19] Müller, M.; Heidelberger, B.; Hennix, M.; Ratcliff, J. Position based dynamics. *Journal of Visual Communication and Image Representation* Vol. 18, No. 2, 109–118, 2007.

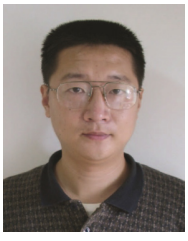
[20] Kovalsky, S. Z.; Aigerman, N.; Basri, R.; Lipman, Y. Controlling singular values with semidefinite programming. *ACM Transactions on Graphics* Vol. 33, No. 4, Article No. 68, 2014.

[21] Andersen, E. D.; Andersen, K. D. The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In: *Applied Optimization, Vol. 33*. Frenk, H.; Roos, K.; Terlaky, T.; Zhang, S. Eds. Springer US, 197–232, 2000.

Zhendong Wang is a Ph.D. candidate in the College of Computer Science and Technology at Zhejiang University, Hangzhou, China. He received his B.S. degree in 2013 from the College of Computer at Wuhan University, Wuhan, China. His research interests include physically-based cloth simulation, collision detection, and GPU computing.

Tongtong Wang is a Ph.D. candidate in the College of Computer Science and Technology at Zhejiang University, Hangzhou, China. She received her B.S. degree in 2014 from Shandong University, Jinan, China. Her research interests include cloth simulation and collision detection.

Min Tang received his B.S. degree in 1994 and Ph.D. degree in 1999 from the Department of Computer Science and Engineering at Zhejiang University, Hangzhou, China. Currently, he is a professor at Zhejiang University. He was a visiting scholar in the Department of Computer Science, Wichita State University in 2006, and in the Department of Computer Science, UNC-Chapel Hill in 2008. His research interests include image processing, CAD&CG, and collision detection.



Ruofeng Tong received his B.S. degree in 1991 from the Department of Mathematics at Fudan University and Ph.D. degree in 1996 from the Department of Mathematics at Zhejiang University, China. Currently, he is a professor in the College of Computer Science and Engineering, Zhejiang University. His research interests include computer vision, image processing, and CAD&CG.