Eulerian-on-Lagrangian Cloth Simulation

NICHOLAS J. WEIDNER, Texas A&M University
KYLE PIDDINGTON, California Polytechnic State University
DAVID I.W. LEVIN, The University of Toronto
SHINJIRO SUEDA, Texas A&M University

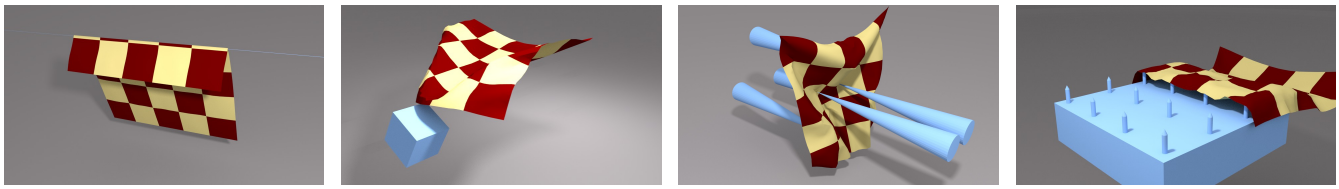


Fig. 1. Still frames from our Eulerian-on-Lagrangian cloth simulations. With our framework, we are able to simulate the smooth sliding of cloth over sharp features such as wires, moving boxes, sharp teeth, and nails.

We resolve the longstanding problem of simulating the contact-mediated interaction of cloth and sharp geometric features by introducing an Eulerian-on-Lagrangian (EOL) approach to cloth simulation. Unlike traditional Lagrangian approaches to cloth simulation, our EOL approach permits bending exactly at and sliding over sharp edges, avoiding parasitic locking caused by over-constraining contact constraints. Wherever the cloth is in contact with sharp features, we insert EOL vertices into the cloth, while the rest of the cloth is simulated in the standard Lagrangian fashion. Our algorithm manifests as new equations of motion for EOL vertices, a contact-conforming remesher, and a set of simple constraint assignment rules, all of which can be incorporated into existing state-of-the-art cloth simulators to enable smooth, inequality-constrained contact between cloth and objects in the world.

CCS Concepts: • **Computing methodologies → Physical simulation**;

Additional Key Words and Phrases: Physical simulation, Lagrangian mechanics, Cloth, Eulerian, Constraints, Contact

ACM Reference Format:
Nicholas J. Weidner, Kyle Piddington, David I.W. Levin, and Shinjiro Sueda. 2018. Eulerian-on-Lagrangian Cloth Simulation. *ACM Trans. Graph.* 37, 4, Article 50 (August 2018), 11 pages. https://doi.org/10.1145/3197517.3201281

1 INTRODUCTION

Cloth simulation has a long history in computer graphics. Starting with the work by Terzopoulos et al. [1987], cloth simulation algorithms have steadily evolved from research curiosities to an integral part of the visual effects pipeline. In fact, cloth simulation has proven to be one of the major success stories for physics-based animation as its phenomenological behavior is difficult to capture for an artist and far simpler to describe using the laws of physics. This allows artist-controlled inputs, such as geometry and initial conditions, to generate life-like results with much less human effort. Motivated by this, research over the last few decades has focused on improving both the performance and physical realism of cloth-simulation.

Due to its highly-deformable nature, cloth undergoes complex colliding interactions with other geometry and itself. Therefore, it is no surprise that much of the research in the field focuses on resolving these collisions. Despite this concentration of effort, there is one crucial scenario which, so far, has yet to be treated: the interaction of cloth with sharp geometric features. Such interactions are common when cloth interacts with the world, from a table cloth pulled over a table edge, to a sheet dragged off a clothing line to the unveiling of a sculpture. Yet this literal (but not figurative) edge case still baffles today's state-of-the-art approaches.

The difficulty arises because all previous cloth simulation algorithms rely on a Lagrangian discretization of the cloth, which only permits bending along its edges. As a pedagogical example, imagine draping the cloth over a sharp edge of a table. Unless the cloth mesh has edges exactly aligned with the table edge, the cloth will not be able to bend sharply, leading to unappealing visual artifacts. For a stationary piece of cloth, this can be resolved by remeshing the cloth, inserting edges that directly align with the table edge. However, what happens if we pull the cloth? We suffer from unacceptable jitter since the cloth can only be remeshed between time steps. This occurs even if we remesh the cloth during the sliding motion, as shown in Fig. 2. In certain scenarios we may even experience catastrophic locking, wherein the cloth, unable to slide over the edge, simply gets stuck. Today, standard approaches to fixing this problem involve perturbing the underlying geometry by approximating the underlying surface as smooth (*e.g.,* by averaging surface normals). However, this approach is unsatisfactory because it prevents us from visually capturing the correct physical sliding behavior of the cloth over the sharp feature, even if this is precisely what an artist wants to achieve. This goes against the very philosophy of physics-based

Authors' addresses: Nicholas J. Weidner, Department of Computer Science & Engineering, Texas A&M University, College Station, TX, weidnerjn@tamu.edu; Kyle Piddington, Department of Computer Science & Software Engineering, California Polytechnic State University, San Luis Obispo, CA, kpiddy@gmail.com; David I.W. Levin, Department of Computer Science, The University of Toronto, Toronto, ON, diwlevin@cs.toronto.edu; Shinjiro Sueda, Department of Computer Science & Engineering, Texas A&M University, College Station, TX, sueda@tamu.edu.

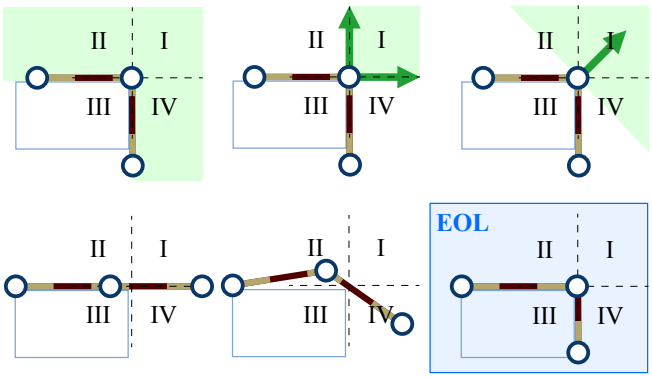


Fig. 2. (Top) Velocity constraints. Left: ignoring neighboring elements, the colliding vertex should be able to move into quadrants I, II, and IV. Center: if we use both contact normals, the strand gets stuck and cannot be pulled left or down. Right: if we use an averaged contact normal, the strand would lift off of the table unexpectedly. (Bottom left & center) If we pull the strand to the left, the vertices would need to jump to remain collision free. (Bottom right) With the EOL approach, the center node does not need to move but can let the material pass through.

animation as we are unintentionally adding counterintuitive, non-physical behavior to our simulations which can prevent achieving a desired physical effect.

In this paper, we propose the first Eulerian-on-Lagrangian [Fan et al. 2013; Li et al. 2013; Sueda et al. 2011] cloth framework—an augmentation of Lagrangian cloth solvers that allows cloth to behave correctly when interacting with both smooth and sharp features based solely on the input geometry. We develop a new contact-conforming remeshing algorithm and propose a new set of contact constraints that, together, correctly handle sliding contact at sharp edges and corners by ensuring degrees of freedom (DOFs) exist at contact boundaries and automatically allowing a contact solver to apportion motion between Lagrangian and Eulerian DOFs in a principled way. These contributions mean that our method avoids constraint jitter and catastrophic locking due to contact (Fig. 2). Furthermore, because our technique works in conjunction with, rather than supplanting, Lagrangian simulation schemes, our algorithm gracefully elides to standard cloth simulation in all other cases. We demonstrate the efficacy and applicability of our method by integrating our framework with the state-of-the-art cloth remesher found in ARCSim [Narain et al. 2012] and using it to resolve a number of challenging scenarios involving close contact of cloth with sharp geometric features.

2 RELATED WORK

Cloth simulation has received tremendous amount of attention from the graphics community, starting with the seminal work by Terzopoulos et al. [1987]. Here we focus on works that are most relevant to our work.

Considerable amount of work has focused on improving the efficiency of cloth simulators, using, for example: linearly implicit integration [Baraff and Witkin 1998]; implicit-explicit integration [Boxerman and Ascher 2004]; subspace integration [Hahn et al. 2014]; and multigrid [Tamstorf et al. 2015]. Some works are focused on optimizing the cloth behavior and collisions specifically for character animation [Cordier and Magnenat-Thalmann 2002, 2005; Kim et al. 2012; Xu et al. 2014]. There have also been numerous works on improving the mechanical behavior and material models of the cloth [Bergou et al. 2006; Provot 1996; Tamstorf and Grinspun 2013; Thomaszewski et al. 2009, 2006; Volino et al. 2009; Wardetzky et al. 2007] and data-driven materials [Bhat et al. 2003; Miguel et al. 2012, 2013; Wang et al. 2011]. Some have focused on simulating or adding wrinkles and other high-frequency details [Bridson et al. 2003; Chen et al. 2013; Gillette et al. 2015; Hadap et al. 1999; Kavan et al. 2011; Müller and Chentanez 2010; Pabst et al. 2008; Rémillard and Kry 2013; Rohmer et al. 2010; Wang et al. 2010]. Recently, some works have focused on: interactive editing [Umetani et al. 2011]; precomputation [Kim et al. 2013]; and simulating individual yarns [Cirio et al. 2014, 2015; Kaldor et al. 2008, 2010]. Creasing and adaptive remeshing for cloth and paper has also garnered much attention [Bender and Deul 2013; Koh et al. 2014; Narain et al. 2013, 2012; Patkar et al. 2015; Pfaff et al. 2014; Schreck et al. 2015; Villard and Borouchaki 2005].

Perhaps the most important topic for cloth simulation is collision handling. Indeed, from the early days of cloth animation, much of the attention has been focused on resolving collisions, using, for example: collision zones and consistency checking [Volino et al. 1995]; hybrid constraint forces [Volino and Magnenat-Thalmann 2000]; rigid impact zones [Bridson et al. 2002; Provot 1997]; tangle removal as a post-process [Baraff et al. 2003]; constraint projection [Goldenthal et al. 2007]; inelastic projection [Harmon et al. 2008]; globally coupled impulses [Sifakis et al. 2008]; asynchronous variational integrators and nested barrier potentials [Ainsley et al. 2012; Harmon et al. 2009]; and air meshes [Müller et al. 2015]. None of these previous works can handle sliding around sharp features due to the *fundamental limitations of the Lagrangian discretization.*

Several previous works attempt to overcome the limitations of purely Lagrangian approaches. The most famous is the Arbitrary Lagrangian-Eulerian (ALE) method, which introduces an additional computational domain, the reference domain, to the standard continuum mechanics picture, which includes the material domain and the spatial domain [Belytschko et al. 2013]. This extra domain allows independent movement of the simulation mesh with respect to the material it is tracking. ALE has been used to great effect for simulating complex effects such as solid fluid coupling as well as for simple contacting scenario for 2D or 3D objects [Sarrate et al. 2001]. Typically, ALE mesh movement relies on defining an additional interpolation function or energy which drives mesh movement [Sarrate et al. 2001]. This is necessary because introducing referential DOFs creates a singularity in the motion description which must be resolved [Fan et al. 2013]. The Eulerian-on-Lagrangian (EOL) method [Fan et al. 2013; Sueda et al. 2011] is a close cousin to ALE with three important differences. First, EOL does not rely on a referential domain, instead chaining together Eulerian or Lagrangian domains into kinematic hierarchies; second, EOL can work with generalized DOFs; and third, EOL methods do not rely on additional functions to determine mesh movement, instead solving for it simultaneously as a function of the physics of the simulated system. Such techniques also require dealing with the inherent motion singularity and the manner in which this is accomplished is part of EOL

algorithm design, be it using a least-squares partition of velocities [Fan et al. 2013] or a reduced coordinate / constraint based approach [Cirio et al. 2014, 2015; Li et al. 2013; Sachdeva et al. 2015; Sueda et al. 2011].

In this paper, we follow an EOL approach and make several important contributions. First, we extend EOL to the case of general cloth simulation, which is an object with a 2D material domain in 3D world space. This is unique to all previous ALE and EOL methods. (Skin simulation work by Li et al. [2013] did not have Lagrangian DOFs; Yarn-level simulation work by Cirio et al. [2014, 2015] did not have a 2D material domain.) Second, we develop a simple set of rules that allow for automatic computation of Eulerian and Lagrangian motions. Third, we extend a state-of-the-art remesher (ARCSim) [Narain et al. 2012] to allow conformal remeshing. Finally, our formulation works with contact inequality constraints, allowing not just sliding, but separation of contacting objects, something which is yet to be demonstrated for previous ALE or EOL methods.

3 OVERVIEW

At each vertex i of the mesh, we store the Lagrangian DOF, $\boldsymbol{x}_i \in \mathbb{R}^3$, and the Eulerian DOF, $\boldsymbol{X}_i \in \mathbb{R}^2$. The Lagrangian DOF represents the world coordinates of the vertex, and the Eulerian DOF represents the material coordinates of the vertex. (One interpretation is that the Eulerian DOFs represent the vertex texture coordinates, and a texel represents a cloth material point.) In traditional purely Lagrangian methods, the material coordinates of all vertices are fixed, whereas in our method, the material coordinates of *some* vertices can vary over time. We call such vertices EOL vertices, and we use these where the cloth is in contact with sharp geometric features. The rest of the cloth is discretized with standard Lagrangian vertices. We follow the standard notation and use $\boldsymbol{q}_i$ to denote the full DOF of a vertex. For a Lagrangian vertex, $\boldsymbol{q}_i = \boldsymbol{x}_i \in \mathbb{R}^3$, whereas for an EOL vertex, $\boldsymbol{q}_i = (\boldsymbol{x}_i \quad \boldsymbol{X}_i)^T \in \mathbb{R}^5$.

We use these EOL vertices wherever the cloth is in contact with sharp edges or corners of another body. For example, if the cloth is in contact with a box, box-face vs. cloth-vertex collisions are handled using standard Lagrangian approaches, whereas box-edge vs. cloth-edge and box-vertex vs. cloth-face collisions are handled using our EOL framework. The border of the cloth must be handled in a special way: cloth corner vertices are always purely Lagrangian, and cloth edge vertices are Eulerian only in the direction along the edge of the cloth. (Using the texture mapping interpretation from above, these conditions imply that the texture of the cloth cannot slide outside of the border of the cloth.) For clarity of exposition, we will assume that all non-cloth bodies are rigid *boxes*, though any rigid/deformable body would work well with our method, as long as there is a way to distinguish between hard and soft edges.

Alg. 1 shows the procedure for taking a time step. In the rest of this paper, we will go over the important steps of this time stepper. We will first present the core of our framework—EOL cloth dynamics in §4. Then, we will describe the set of simple geometric rules for constructing the constraints on the Lagrangian and Eulerian velocities in §5. Finally, we will go over our conformal remeshing solution in §6.

Algorithm 1 EOL Cloth Time Stepper

```
while simulating do
    Detect collisions                          ▷ External call
    Preprocess & Remesh                        ▷ Section 6
    Compute EOL constraints on new mesh        ▷ Section 5
    Integrate velocities and positions         ▷ Section 4
end while
```

4 EOL CLOTH DYNAMICS

We discretize the cloth with triangles that conform to the hard edges obtained from the collision detector. Let the three vertices of a triangle be denoted (a, b, c), and $\boldsymbol{X} \in \mathbb{R}^2$ be an arbitrary material point inside this triangle. From the Eulerian DOFs of the triangle $(\boldsymbol{X}_a, \boldsymbol{X}_b, \boldsymbol{X}_c)$ we can compute the barycentric coordinates (α, β, γ) of this material point, $\boldsymbol{X}$, using the standard expression for barycentric coordinates (Eq. A.3). The world position, $\boldsymbol{x} \in \mathbb{R}^3$, corresponding to this material point can then be computed as

$$\boldsymbol{x}(\boldsymbol{X}) = \alpha \boldsymbol{x}_a + \beta \boldsymbol{x}_b + \gamma \boldsymbol{x}_c, \tag{1}$$

where $(\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c)$ are the Lagrangian DOFs of the triangle. The world position is not only a function of the Lagrangian DOFs of the triangle but also of the Eulerian DOFs, since the barycentric coordinates depend on these Eulerian DOFs. Texture mapping again gives us a useful analogy for intuition. Even if we keep the nodal positions (Lagrangian DOFs) fixed, if we modify the nodal texture coordinates (Eulerian DOFs), the cloth moves in world space.

This dependence of the barycentric coordinates on the Eulerian DOFs becomes important when we take the time derivative of Eq. 1, since we need to account for the time derivative of the Eulerian coordinates as well. After some rearranging (see Appendix A for the derivation), we obtain

$$\dot{\boldsymbol{x}} = (\alpha \dot{\boldsymbol{x}}_a + \beta \dot{\boldsymbol{x}}_b + \gamma \dot{\boldsymbol{x}}_c) - F(\alpha \dot{\boldsymbol{X}}_a + \beta \dot{\boldsymbol{X}}_b + \gamma \dot{\boldsymbol{X}}_c), \tag{2}$$

where $F \in \mathbb{R}^{3\times 2}$ is the deformation gradient of the triangle:

$$\begin{aligned} F &= D_x D_X^{-1} \\ D_x &= \begin{pmatrix} \boldsymbol{x}_b - \boldsymbol{x}_a & \boldsymbol{x}_c - \boldsymbol{x}_a \end{pmatrix} \\ D_X &= \begin{pmatrix} \boldsymbol{X}_b - \boldsymbol{X}_a & \boldsymbol{X}_c - \boldsymbol{X}_a \end{pmatrix}. \end{aligned} \tag{3}$$

Here, $D_x \in \mathbb{R}^{3\times 2}$ and $D_X \in \mathbb{R}^{2\times 2}$ are the matrices constructed from the edge vectors of the triangle. The appearance of the deformation gradient here should not surprise us, since its purpose is to map deformations from material space to world space. The negative sign is due to the derivative of the barycentric coordinates with respect to the Eulerian DOFs. This implies that when we change the Eulerian DOF of a node, the cloth material moves in the opposite direction, just like how the motion of texture is the opposite to the motion of texture coordinates.

Finally, for any material point, $\boldsymbol{X}$, inside a triangle, the Jacobian, $\mathrm{J} \in \mathbb{R}^{3\times 15}$, for mapping the generalized velocities of the three vertices of the triangle to the world velocity of the material point is

$$\mathrm{J} = \begin{pmatrix} \alpha I & \beta I & \gamma I & -\alpha F & -\beta F & -\gamma F \end{pmatrix}, \tag{4}$$

where I is the 3×3 identity matrix, and F is the deformation gradient from Eq. 3. The world space velocity of a material point is then $\dot{\boldsymbol{x}} =$

$\mathsf{J}\dot{\mathsf{q}}$, where $\dot{\mathsf{q}} = (\dot{\boldsymbol{x}}_a\ \dot{\boldsymbol{x}}_b\ \dot{\boldsymbol{x}}_c\ \dot{\boldsymbol{X}}_a\ \dot{\boldsymbol{X}}_b\ \dot{\boldsymbol{X}}_c)^T \in \mathbb{R}^{15}$ is the concatenation of the Lagrangian and Eulerian DOFs of the triangle. The transpose of the Jacobian maps a world force to a generalized force: $\mathsf{f} = \mathsf{J}^T \boldsymbol{f}$. If the material point happens to be at a triangle node, the Jacobian simplifies to $\mathsf{J} = (I\ \ -F) \in \mathbb{R}^{3\times 5}$.

4.1 Generalized Inertia

The kinetic energy of a triangle can be expressed as

$$T = \frac{1}{2}\int_A \rho\, \dot{\boldsymbol{x}}^T \dot{\boldsymbol{x}}\, dA, \tag{5}$$

where ρ is the area density, and the integral is over the area of the triangle in material space spanned by $(\boldsymbol{X}_a, \boldsymbol{X}_b, \boldsymbol{X}_c)$. Using α and β as the variables of integration over the triangle, we obtain

$$T = \frac{1}{2}\int_{\alpha=0}^{1}\int_{\beta=0}^{1-\alpha} \rho\, \dot{\boldsymbol{x}}^T \dot{\boldsymbol{x}}\, A\, d\beta\, d\alpha, \tag{6}$$

where $A = |\det(D_X)|/2$ is the area of the triangle in material space. Integrating out α and β and rearranging the terms, we arrive at

$$T = \frac{1}{2}\dot{\mathsf{q}}^T \mathsf{M}\dot{\mathsf{q}}, \tag{7}$$

where M is the generalized inertia, obtained by using the Jacobian from Eq. 4 and then integrating the result:

$$\mathsf{M} = \frac{\rho A}{12}\begin{pmatrix} 2I & I & I & -2F & -F & -F \\ \cdot & 2I & I & -F & -2F & -F \\ \cdot & \cdot & 2I & -F & -F & -2F \\ \cdot & \cdot & \cdot & 2F^TF & F^TF & F^TF \\ \cdot & \cdot & \cdot & \cdot & 2F^TF & F^TF \\ \cdot & \cdot & \cdot & \cdot & \cdot & 2F^TF \end{pmatrix}, \tag{8}$$

where the dots indicate symmetric terms. This generalized inertia matrix is 15×15, corresponding to the 9 Lagrangian DOFs and 6 Eulerian DOFs of the triangle. The upper left 3×3 blocks of the inertia matrix correspond to the Lagrangian DOFs and are constant over time, an advantage exploited by Lagrangian simulators. The rest of the inertia matrix must be computed at every time step, since F is a function of both Lagrangian and Eulerian DOFs.

4.2 Generalized Forces

The EOL framework works with any set of forces. For each node, we compute its world force, $\boldsymbol{f}$, and then use the Jacobian transpose from Eq. 4 to map this world force into its Lagrangian and Eulerian force components:

$$\begin{pmatrix} \boldsymbol{f}^L \\ \boldsymbol{f}^E \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ -F^T\boldsymbol{f} \end{pmatrix}. \tag{9}$$

The corresponding stiffness matrix, K, for a node is similarly transformed into its Lagrangian and Eulerian components as

$$\begin{pmatrix} K^{LL} & K^{LE} \\ K^{EL} & K^{EE} \end{pmatrix} = \begin{pmatrix} K & -KF \\ -F^TK & F^TKF \end{pmatrix}. \tag{10}$$

In our current implementation, we use the corotated linear material model for membrane forces [Bender and Deul 2013; Etzmuß et al. 2003; Thomaszewski et al. 2009] and discrete Willmore energy for bending forces [Tamstorf and Grinspun 2013; Wardetzky et al. 2007]. Whenever we need the deformation gradient at a vertex, we simply take an element-wise average. An alternative approach would be to take the polar decomposition to interpolate the rotation separately.

4.3 Equations of Motion

In our current implementation, we use the linearly implicit integration scheme at the velocity level, popularized by the work on efficient cloth simulation by Baraff and Witkin [1998]:

$$\left(\mathsf{M} - h^2\mathsf{K}\right)\dot{\mathsf{q}}^{(k+1)} = \mathsf{M}\dot{\mathsf{q}}^{(k)} + h\mathsf{f}, \tag{11}$$

where h is the time step, and the superscript k indicates the current time step. The EOL framework is not tied to a specific integration scheme, and should work equally well with other schemes.

Adding the EOL equality and inequality constraints, which are described later in §5, and applying Gauss's Principle of Least Constraint [Lanczos 1986], we arrive at the following, which is solved for the new velocities, $\dot{\mathsf{q}}^{(k+1)}$.

$$\begin{aligned} \underset{\dot{\mathsf{q}}}{\text{minimize}} \quad & \frac{1}{2}\dot{\mathsf{q}}^T\tilde{\mathsf{M}}\dot{\mathsf{q}} - \dot{\mathsf{q}}^T\tilde{\mathsf{f}} \\ \text{subject to} \quad & \mathsf{A}_{\text{eq}}\dot{\mathsf{q}} = 0 \\ & \mathsf{A}_{\text{ineq}}\dot{\mathsf{q}} \geq 0, \end{aligned} \tag{12}$$

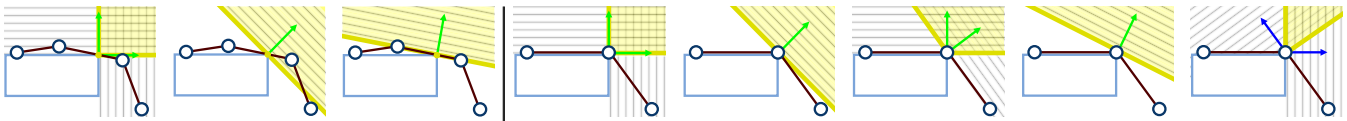
where $\tilde{\mathsf{M}} = \mathsf{M} - h^2\mathsf{K}$, and $\tilde{\mathsf{f}} = \mathsf{M}\dot{\mathsf{q}}^{(k)} + h\mathsf{f}$. Finally, the Lagrangian and Eulerian DOFs are both updated as $\mathsf{q}^{(k+1)} = \mathsf{q}^{(k)} + h\dot{\mathsf{q}}^{(k+1)}$.

5 EOL CLOTH CONSTRAINTS

Before we describe the constraints we apply to the EOL vertices (§5.2), we first review how one would constrain a Lagrangian cloth in contact with sharp geometric features (§5.1). As we saw in Fig. 2, there are difficulties in dealing with position-level as well as velocity-level constraints. Here, however, we expand only on velocity-level constraint problems.

5.1 Constraints for Lagrangian Cloth

Let us consider a 1D strand for illustration. Without conformal remeshing, the strand contacts the box corner at some element, as



(a) Box normals (b) Box avg normal (c) Strand normal (d) Box normals (e) Box avg normal (f) Strand normals (g) Strand avg normal (h) Strand binormals

Fig. 3. Possible contact constraints for a strand-box collision without conformal remeshing (a-c) and with conformal meshing (d-h). (a) If we use both normals of the box corner, the strand would be over constrained. (b & c) If we use the averaged box normal or the strand normal, it would result in an unnatural lift off. (d-g) Similar choices exist with conformal remeshing.

shown in Fig. 3(a-c). There are two normals at the box corner, as shown in (a), as well as the single normal from the strand element, as shown in (c). If we choose (a), then the strand becomes over-constrained and locked, unable to move left or down. Instead, we must average the box normals (b) or use the strand normal (c). Unfortunately, this causes the strand to lift off unnaturally, since the constrained point must stay in the positive halfspace of the constraint.

If we apply conformal remeshing, as shown in Fig. 3(d-h), then there are now two normals to choose from the two neighboring strand elements (f). Using these two strand normals still results in an over-constrained configuration, since it prevents the colliding node to go below the top of the box. A Lagrangian simulator must still use averaged normals (e) or (g), since otherwise the strand becomes over-constrained. Unfortunately, with (e) or (g), the cloth is now under-constrained, as these constraints may allow the cloth to penetrate the box. To summarize, conformal remeshing helps a Lagrangian simulator when the cloth is static, but it does not resolve the problem stemming from sliding motion.

5.2 Constraints for EOL Cloth

With the EOL approach, we get around this problem by using the Eulerian DOFs to move the cloth around the sharp features.

Before we describe how we construct the Lagrangian and Eulerian constraints, we first mention an important consideration when dealing with EOL methods: the inertia matrix in Eq. 8 can become singular depending on the configuration of the cloth. This is because for some Eulerian velocities, there may be a corresponding Lagrangian velocity that exactly cancels out the motion of the cloth material. As an illustration, let us assume that an undeformed cloth is laid flat on the X-Y plane, and that there is an EOL vertex in the middle of the cloth. In this configuration, we can move the Lagrangian DOFs of the vertex (*i.e.,* vertex position) in one direction and the Eulerian DOFs (*i.e.,* vertex texture coordinates) in the other direction[1] so that the actual cloth does not move in world space. Thus, this combination of Lagrangian and Eulerian velocities lies in the nullspace of the inertia matrix. One potential approach for dealing with this singularity is to use least squares to solve for the largest Lagrangian velocities first [Fan et al. 2013; Malgat et al. 2015]. In our setting of cloth simulation, however, we can exactly account for the singularity by using the local geometry of the cloth and the box. In the rest of this section, we describe a simple set of rules for constructing these constraints on the Lagrangian and Eulerian velocities of an EOL vertex.

There are two cases we must consider: contact with box corner and contact with box edge, which are discussed in the following two paragraphs. In both cases, we use the box normal cone (3D analogues of Fig. 3(d)). Alg. 2 summarizes the procedure for constructing these constraints on the EOL vertices.

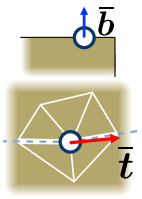
Corner: The Lagrangian constraint for a corner EOL vertex is constructed from an orthonormal frame at the corner of the box. Let the box normals be denoted $\boldsymbol{n}_1$, $\boldsymbol{n}_2$, and $\boldsymbol{n}_3$. The Lagrangian velocity constraint is then $\boldsymbol{n}_1^T\dot{\boldsymbol{x}} \geq 0$, $\boldsymbol{n}_2^T\dot{\boldsymbol{x}} \geq 0$, and $\boldsymbol{n}_3^T\dot{\boldsymbol{x}} \geq 0$. We do not need any constraints on the Eulerian velocity for a corner EOL vertex, since we use these Eulerian DOFs to allow the cloth to slide freely around the corner.

[1] Note the negative sign in Eq. 2 (derivation in Appendix A).

Algorithm 2 EOL Constraint Generation

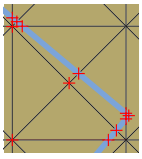
```
for each EOL vertex v do
    if v colliding with box corner then
        Lagrangian Constraint: n1^T ẋ ≥ 0, n2^T ẋ ≥ 0, n3^T ẋ ≥ 0
        Eulerian Constraint: none
    else if v colliding with box edge then
        Lagrangian Constraint: n1^T ẋ ≥ 0, n2^T ẋ ≥ 0
        if v on cloth border then
            Eulerian Constraint: b̄^T Ẋ = 0
        else
            Eulerian Constraint: t̄^T Ẋ = 0
        end if
    end if
end for
```

Edge: The Lagrangian constraint for an edge EOL vertex is constructed from the box normals, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$. We want the vertex to be able to move freely (in the Lagrangian sense) along the box tangent but be able to lift off if necessary. The Lagrangian velocity constraint is then $\boldsymbol{n}_1^T\dot{\boldsymbol{x}} \geq 0$ and $\boldsymbol{n}_2^T\dot{\boldsymbol{x}} \geq 0$. The Eulerian velocity constraint depends on whether the vertex is on the cloth border or not. For EOL vertices on the cloth border, the Eulerian constraint is $\bar{\boldsymbol{b}}^T\dot{\boldsymbol{X}} = 0$, where $\bar{\boldsymbol{b}}$ is the vector orthogonal the cloth border in material space (top inset figure). This constraint ensures that the cloth material remains affixed to the border. For internal EOL vertices, the Eulerian constraint is $\bar{\boldsymbol{t}}^T\dot{\boldsymbol{X}} = 0$, where $\bar{\boldsymbol{t}}$ is the averaged material space tangent constructed from the two edges of the colliding vertex that are lying on the box edge. (In the bottom inset figure, the box collision in material space is shown in dotted blue, and the two edges are shown in thick white.) This constraint ensures that any motion of the cloth along the box edge is realized by the Lagrangian DOF rather than the Eulerian DOF.



All of the local constraints described in this section are collected into global matrices so that the constraints can be written as $A_{eq}\dot{q} = 0$ and $A_{ineq}\dot{q} \geq 0$ where $\dot{q}$ is the concatenation of all nodal Lagrangian and Eulerian velocities.

6 CONFORMAL REMESHING



To remesh the cloth, we use ARCSim (v0.3.1) [Narain et al. 2013, 2012; Pfaff et al. 2014], which has curvature based metrics to help avoid visually unappealing changes to the triangle mesh. In the inset figure, we show a typical remeshing scenario when the cloth first make contact with the box. The cloth mesh is shown in black, the box mesh is shown in blue, and the collisions are shown in red. Our goal is to remesh the cloth so that it conforms to the box. Even though ARCSim has the ability to "preserve" certain edges during triangulation, it does not work out-of-the-box for conformal remeshing for two reasons:

- Our preserved edges move around in the material domain, potentially creating extremely thin triangles.
- Collisions often occur very close to each other. Simply marking all collisions as preserved (*e.g.,* all red crosses touching the blue box mesh in the inset figure) does not allow ARCSim enough freedom to remesh properly.

Therefore, we preprocess the mesh before we send the mesh to ARCSim. This preprocessing procedure is applicable to all conformal remeshing, either with standard Lagrangian or with our EOL framework.

6.1 Preprocessing

Before remeshing with ARCSim, we first scan a list of collision features (*e.g.,* box edges and box corners), along with a list of existing conformal vertices. (*Conformal* vertices and edges refer to cloth vertices and edges that are in contact with sharp features. In our EOL framework, conformal vertices are our EOL vertices.) We insert new conformal vertices into our mesh either by splitting faces or edges of the mesh at untracked features. We then sort and connect the conformal vertices to form conformal edges. Then we repeat the following steps, iterating through all triangles incident to at least one conformal vertex, until no more changes are made. In these steps, when we collapse an edge between a conformal vertex and a non-conformal vertex, we always collapse toward the conformal vertex.

- Collapse non-conformal edges that are below a threshold. (For our examples, we used 1% of the cloth length as our threshold.)
- Collapse conformal edges that are below a threshold. The edge can be collapsed in either direction unless one of the conformal vertices is a cloth border/corner, which must be preserved.
- For an ill-conditioned triangle, we split one of its edges to insert a new vertex, which will then be removed during the next iteration of the preprocessing loop.
  - If it has one conformal vertex, split the edge opposite to it.
  - If it has two conformal vertices, split the conformal edge.
  - If it has three conformal vertices, split the non-conformal edge. We assume that a triangle cannot have three conformal edges—*i.e.,* the cloth triangles are sufficiently small compared to the sharp features.

This preprocessing scheme has worked well for our examples, but we do not have a convergence proof. In some cases, it may not be possible to remove all ill-conditioned triangles while preserving conformal features. In such cases, the time step must be slowed down accordingly. In our experience, EOL simulations produce much fewer ill-conditioned triangles than conformal Lagrangian simulations, making it much more robust.

We do not allow conformal remeshing near the border of the cloth. Specifically, before we insert a conformal edge that touches the cloth border, we check to make sure that the angle between the edge and the border is above a threshold. This prevents the formation of thin triangles incident to cloth borders. In the absence of collisions with sharp features, our algorithm reverts to a standard non-conformal Lagrangian cloth simulation using ARCSim as the remesher.

6.2 Velocity Transfer

Whenever the cloth is remeshed, the velocities must be interpolated at the new vertex positions. There are four ways in which a new vertex can be introduced, and we use the following scheme to compute the new velocity of the newly inserted vertex.

- New Lagrangian vertex inside a Lagrangian triangle. This is the standard case, and we simply use barycentric averaging to compute the vertex's Lagrangian velocity.
- New Lagrangian vertex inside an EOL triangle. If the newly inserted vertex happens to be inside a triangle with one or more EOL vertices, we must first compute the world velocity at these EOL vertices using Eq. 2. Then we use barycentric averaging to compute the inserted vertex's Lagrangian velocity.
- New EOL vertex from an EOL edge split. If the remesher inserts a new EOL vertex by splitting an edge between two EOL vertices, we simply interpolate both the Lagrangian and Eulerian velocities of the two EOL vertices.
- New EOL vertex from collision. Whenever we insert an new EOL vertex as a result of a collision, we first compute the world velocity, $\dot{\boldsymbol{x}}^w$, at the vertex by barycentric averaging. This world velocity is composed of the Lagrangian component, $\dot{\boldsymbol{x}}$, and Eulerian component, $\dot{\boldsymbol{X}}$, and can be expressed as $\dot{\boldsymbol{x}}^w = \dot{\boldsymbol{x}} - F\dot{\boldsymbol{X}}$ (cf. Eq. 2). We put as much of this world velocity into the Eulerian velocity as possible by solving a small constrained optimization problem for $\dot{\boldsymbol{X}}$: min. $\frac{1}{2}\|\dot{\boldsymbol{x}}^w + F\dot{\boldsymbol{X}}\|^2$ s.t. $A_{\text{eq}}\dot{\boldsymbol{X}} = 0$. In other words, we minimize the Lagrangian velocity subject to equality constraints from §5. This turns into a 3×3 linear system:

$$\begin{pmatrix} F^T F & A_{\text{eq}}^T \\ A_{\text{eq}} & 0 \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{X}} \\ \lambda \end{pmatrix} = \begin{pmatrix} -F^T \dot{\boldsymbol{x}}^w \\ 0 \end{pmatrix}. \tag{13}$$

  The Lagrangian velocity is then computed as $\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}^w + F\dot{\boldsymbol{X}}$.

If an EOL vertex lifts off, it becomes a Lagrangian vertex. In this case, the new Lagrangian velocity must take into account the Eulerian velocity from the last step, using Eq. 2.

7 RESULTS

We implemented our system in C++ and ran the simulations on a consumer desktop with an Intel Core i7-7700 CPU @ 3.6 Ghz and 16 GB of RAM. The code is single-threaded and uses Eigen for linear algebra and Mosek for quadratic programs. In Figs. 4 and 6, we show some still frames from the simulations. (Please also see the accompanying video.) In Table 1, we show the performance numbers.

Cloth sliding over box EDGE. In this example, shown in Fig. 4, the cloth is pulled over an edge of a box. For comparison, we also show how Lagrangian cloth simulations behave under the same scenario. Unless otherwise stated, all Lagrangian simulations use averaged constraints (Fig. 3(e)) at sharp features, which approximates the underlying box geometry as smooth.

- With static regular discretization, the cloth is able to form a sharp bend because the box edge happens to be aligned with the cloth mesh. However, bending artifacts become obvious as soon as we pull the cloth.
- With static irregular discretization, the cloth is unable to form a sharp bend.

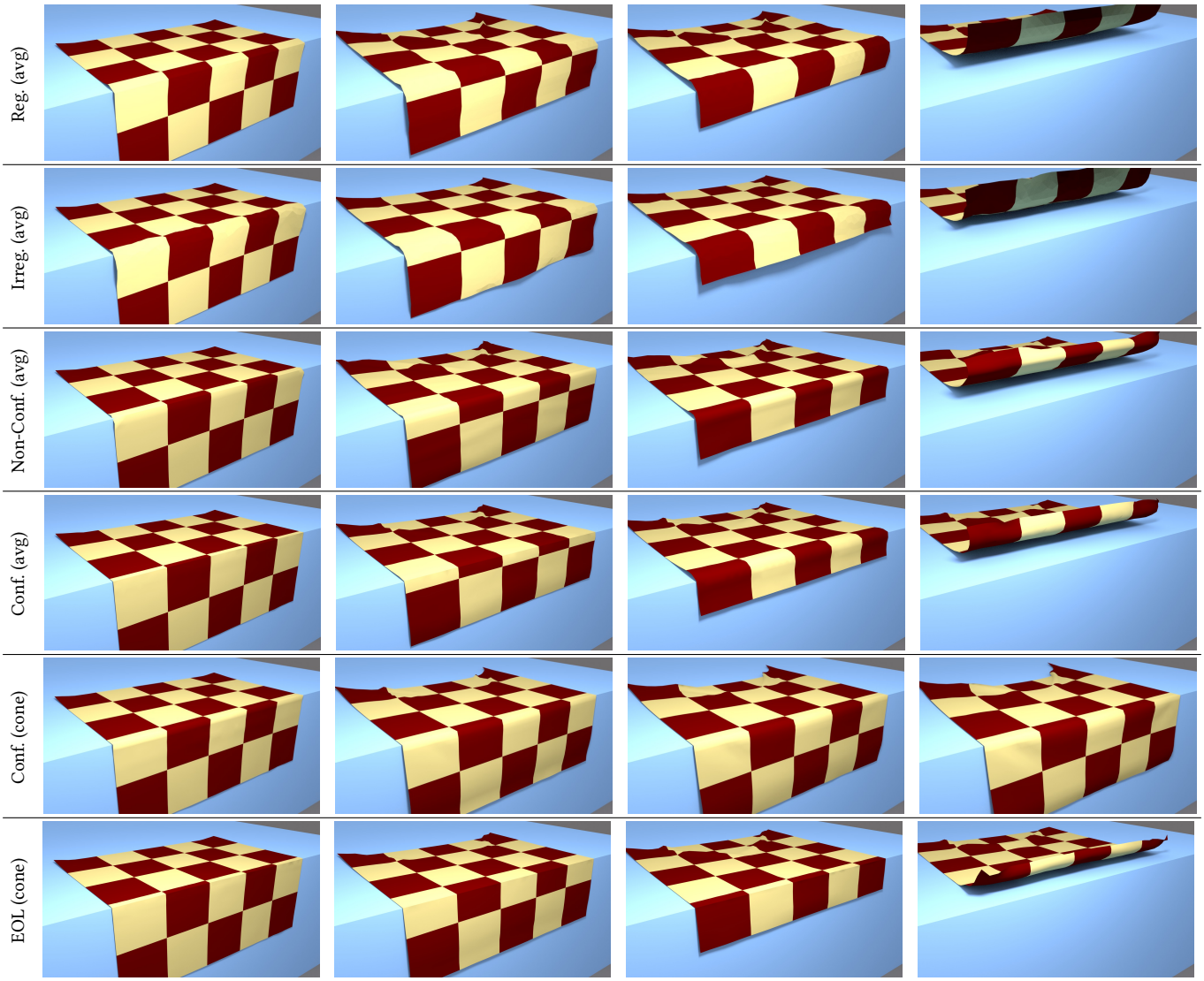


Fig. 4. Still shots from EDGE simulations. From top to bottom: LAG static regular with averaged constraints, LAG static irregular with averaged constraints, LAG non-conformal with averaged constraints, LAG conformal with averaged constraints, LAG conformal with cone constraints, EOL with cone constraints.

- With non-conformal remeshing (*i.e.,* vanilla ARCSim), the cloth is able to bend *semi*-sharply only if we allow ARCSim to generate many triangles. Moreover, when we pull the cloth, the cloth lifts unnaturally because it is not able to bend exactly at the box edge, even at high-resolution. This artificial bending energy is completely independent of the material bending stiffness.
- With naïve conformal remeshing (*i.e.,* vanilla ARCSim with "preserved" edges), the simulation abruptly halts when the cloth hits the box because the collision detector generates too many contact points, which creates too many small "preserved" edges to be passed to ARCSim. See the inset figure in §6.
- With conformal remeshing (*i.e.,* our preprocessing + ARCSim), the cloth is able to bend sharply. When we pull the cloth, however, averaged normal constraints (Fig. 3(e)) cause the cloth to lift unnaturally, at any resolution. Again, this artificial bending energy is independent of the material bending stiffness. Also, because of the Lagrangian vertex motion very close to the sharp features, the collision detector parameters must be highly tuned to detect all the collisions correctly. Furthermore, when the cloth is bent sharply, this liftoff *always* causes the cloth to penetrate the box, which must be projected back. (As an aside, this implies that continuous collision detection cannot be used.)
- With conformal remeshing *and* with proper cone constraints (Fig. 3(d)), the cloth is again able to bend sharply, but as soon as we pull the cloth, it locks catastrophically. (This result is not included in Table 1.)
- With our EOL discretization, the cloth smoothly slides around the edge with a perfectly sharp bend. Because the conformal vertices

(a) (b) (c)

Fig. 5. Lagrangian failure cases for WIRE. (a) Cloth cannot bend sharply. (b) Cloth bunches up and cannot slide over the wire. (c) Cloth falls off the wire.

stay on the box edge, no penetrations are introduced. Also, we are able to use proper cone constraints (Fig. 3(d)) without having to approximating the underlying surface as smooth.

Cloth sliding over a WIRE. Even though conformal Lagrangian simulation works for the EDGE example, when the bending angle becomes more extreme, *it is no longer able to slide smoothly* because with the averaged constraint (Fig. 3(e)), the conformal vertices *can only move horizontally*. Even a small amount of horizontal movement causes large penetrations in the cloth, making subsequent conformal remeshing challenging. Please see Fig. 5 and the video for some of these Lagrangian failure cases. Our EOL cloth, on the other hand, works as expected.

Cloth sliding over NAILS. The EOL framework works just as well with sharp points. In this example, we pull the cloth over a bed of nails. This is an artificially unrealistic scenario, because we would expect the cloth to snag due to the individual cloth fibers getting caught by the nail tips. However, it is important to note that Lagrangian simulations snag due to its discretization rather than by proper physics. With our EOL approach, it would be possible to simulate this snagging behavior properly by including additional external forces.

Cloth sliding through JAWS. We further demonstrate the robustness of our approach by pulling the cloth through "jaws of death." As before, no artificial snagging behavior is observed.

Scripted box PUSH. In this example, we show the cloth being pushed by a scripted box to illustrate how effectively the cloth is able to slide over the box. This example also highlights the proper lift-off of EOL vertices due to our inequality constraints.

Scripted box THROW. This example shows a more dynamic cloth making contact with, sliding over, and then lifting over a scripted box. Many EOL edges and points are created on the fly, as the cloth comes in contact with various edges and corners of the box. Again, no snagging behavior is observed.

Coarse preview with LO-RES cloth. We show that the EOL framework works very well for generating coarse previews of simulations involving sharp features. With a static Lagrangian simulation, we lose the details around the sharp features. With EOL, we are able to retain the sharp features even when the cloth starts to slide off. There are some obvious popping artifacts caused by remeshing along the sharp edges, but this is a side effect of any method that aligns cloth geometry with object geometry, *i.e.,* conformal LAG and EOL, and also occurs at higher resolutions but is less visible. With EOL, we do not get any snagging behavior even at low-res, which, in scenarios where overall quality of motion is paramount, is more important than visual fidelity.

Cloth with FRICTION. In this example, the cloth is dropped between four boxes, hitting their corners. With EOL, the cloth smoothly slides over the corners and edges of the boxes, and without friction, the cloth eventually falls naturally off of the boxes with no bending artifacts or locking. When we repeat the simulation with friction by applying the impulse-based friction formulation of Bridson et al. [2002], the cloth stops rather than falling. For each vertex, we first compute the scalar friction multiplication factor using the world velocity of the vertex (Eq. 2). For Lagrangian vertices, we apply this factor to the tangential component of the velocity as usual. For EOL vertices on a box corner, we apply the factor to just the Eulerian velocity, since the tangential motion is encoded fully by the Eulerian velocity. For EOL vertices on a box edge, we apply the factor to the Eulerian velocity and the tangential component of the Lagrangian velocity (along the box edge).

Table 1. Performance numbers for the examples. #F: Maximum number of faces. %E: Maximum EOL vertex percentage. %CD: Percentage spent in collision detection. %RM: Percentage spent in remesh. %MF: Percentage spent in matrix fill. %VI: Percentage spent in velocity integration (QP). T: Total time per step (ms). Some scenes are run multiple times with different settings. Regular: Lagrangian simulation with a static regular mesh. Irregular: Lagrangian simulation with a static irregular mesh. Non-conformal: Lagrangian simulation with non-conformal remeshing (ARCSim). Conformal: Lagrangian simulation with conformal remeshing (ARCSim with our preprocessing). EOL: Our EOL simulation.

Scene	#F	%E	%CD	%RM	%MF	%VI	T (ms)
EDGE (reg.)	2116	-	1.2	-	6.6	92.1	436.2
EDGE (irreg.)	2000	-	0.9	-	5.0	94.0	589.2
EDGE (non-conf.)	2521	-	0.9	4.7	5.4	88.9	638.9
EDGE (conf.)	2622	-	1.8	5.2	4.5	88.6	729.6
EDGE (EOL)	2971	2.4	0.9	2.5	7.7	88.8	1101.0
WIRE (reg.)	1936	-	0.3	-	3.4	96.3	827.2
WIRE (irreg.)	2055	-	0.2	-	3.3	96.5	896.5
WIRE (non-conf.)	2048	-	2.6	4.1	2.8	96.9	1028.7
WIRE (conf.)	2022	-	0.6	4.0	3.5	91.9	728.3
WIRE (EOL)	2048	2.4	0.2	3.0	7.7	89.1	1058.7
NAILS (EOL)	2012	1.9	0.6	2.9	7.9	88.6	804.2
JAWS (EOL)	3380	0.3	0.3	9.2	21.8	68.7	503.9
PUSH (EOL)	2110	3.1	1.4	6.2	16.7	75.6	454.9
THROW (EOL)	1913	3.9	0.8	3.9	9.0	86.2	568.4
LO-RES (reg.)	256	-	3.2	-	15.8	81.0	18.1
LO-RES (EOL)	133	6.0	16.8	6.2	23.1	53.9	16.3
FRICTION (EOL)	2927	2.9	6.7	8.4	23.6	61.3	451.5

8 CONCLUSION

We introduced a novel Eulerian-on-Lagrangian cloth simulation framework that can robustly simulate the cloth sliding over sharp features, a scenario that cannot be simulated by other methods due to the fundamental limitation of purely Lagrangian simulators. In our framework, we use both Eulerian and Lagrangian DOFs for vertices at the sharp features. We derive the equations of motion for elements that involve these special vertices. We define a simple set of geometric rules for constraining these vertices to remove the redundancy that exists between the Eulerian and Lagrangian

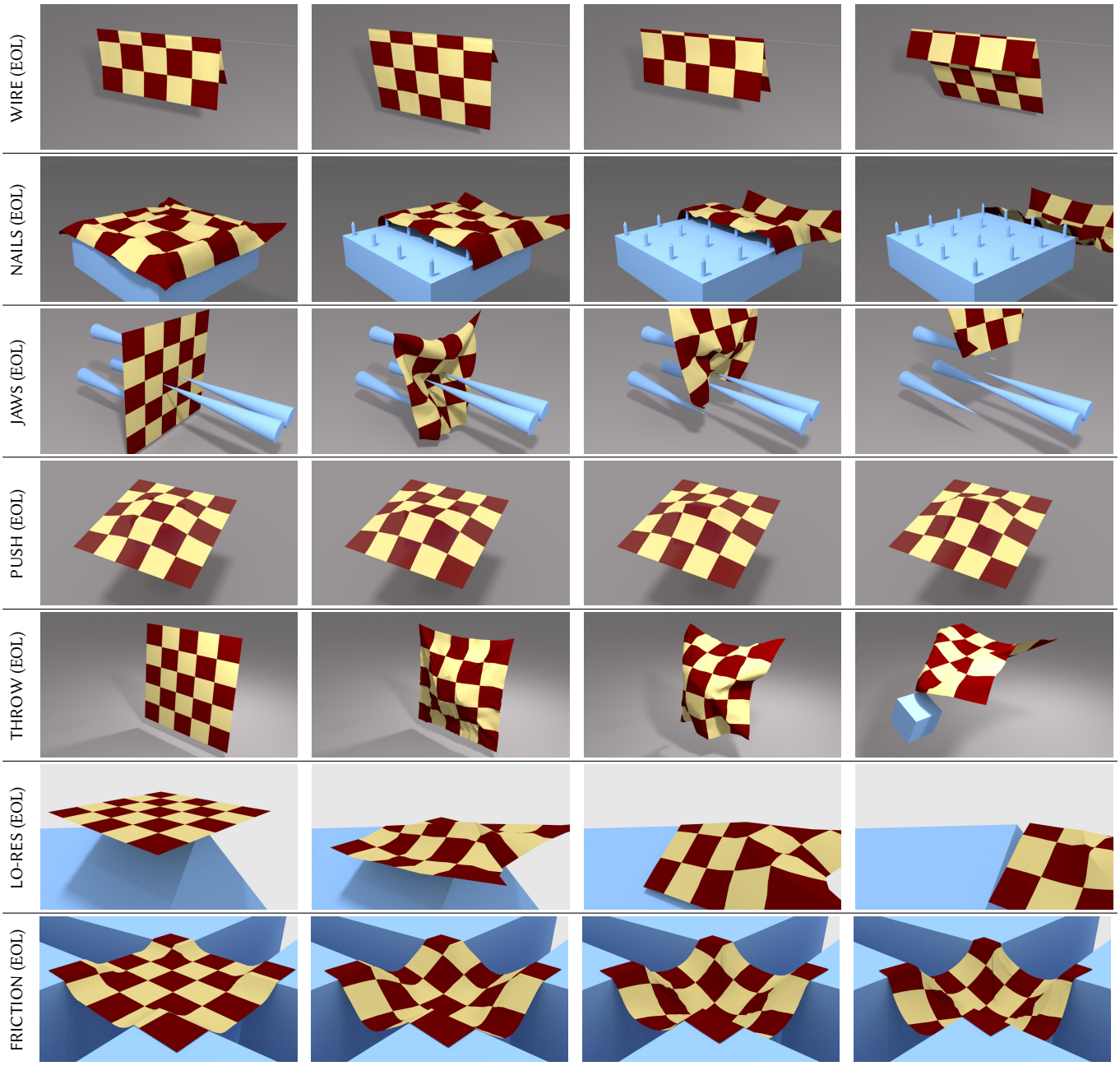


Fig. 6. Still shots from WIRE, NAILS, JAWS, PUSH, THROW, LO-RES, and FRICTION simulations.

DOFs. We extend a state-of-the-art remesher (ARCSim) for conformal remeshing around sharp features. Finally, we show various examples of how our framework is able to handle difficult scenarios involving sliding over sharp edges and corners.

8.1 Future Work

Our work is the first work to use an Eulerian discretization for cloth, and so we hope that it opens many avenues of future work. We have released our source code to encourage future research in Eulerian-on-Lagrangian cloth simulation,[2] some directions of which we discuss next. A limitation common to all adaptive cloth simulators is that the remesher does not take into account the sharp features in the environment. Collision-aware remeshing that avoids interpenetrations would benefit not only our work but all other work on adaptive cloth. Another limitation of our current implementation is that we are computing the per-vertex deformation gradient

[2]https://github.com/sueda/eol-cloth

by element-wise averaging of the incident triangle deformation gradients. We expect to see better energy behavior if we compute this average using Lie algebra. Furthermore, allowing EOL style contact handling for cloth-cloth, cloth-fluid and cloth-deformable body interactions would allow for more seamless simulations of such phenomena. In the current implementation, we simply default to purely Lagrangian handling for any contact which is not between cloth and a static rigid body. Another interesting avenue of future work is to remove the restriction that the border vertices must be Lagrangian (corner nodes cannot be Eulerian; edge nodes can only be Eulerian along the edge tangent). With this modification, we expect to see better transition of EOL to Lagrangian vertices near the border of the cloth. Next, while our work focuses on finite element simulation of cloth, we believe it can be extended to other simulation techniques such as Projective [Bouaziz et al. 2014] and Position-Based Dynamics [Müller et al. 2007]. Finally, even though our approach is the first to enable smooth sliding cloth, it still requires remeshing around sharp features. Removing this dependence on remeshing would improve efficiency and robustness of both conformal Lagrangian and EOL cloth simulations.

ACKNOWLEDGMENTS

We thank Gustavo Lopez for rendering the results. We also thank Zoë Wood, Nikolai Shkurkin, Patrick Riordan, and the anonymous reviewers for their insightful comments. This work was supported in part by NSERC Discovery Grant #1303800 and the Canada Research Chairs Program.

REFERENCES

Samantha Ainsley, Etienne Vouga, Eitan Grinspun, and Rasmus Tamstorf. 2012. Speculative Parallel Asynchronous Contact Mechanics. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 151:1–151:8.

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proc. SIGGRAPH 98, Annual Conference Series*. 43–54.

David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling Cloth. *ACM Trans. Graph.* 22, 3 (July 2003), 862–870.

Ted Belytschko, Wing Kam Liu, Brian Moran, and Khalil Elkhodary. 2013. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons.

Jan Bender and Crispin Deul. 2013. Adaptive cloth simulation using corotational finite elements. *Computers & Graphics* 37, 7 (2013), 820 – 829.

Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. 2006. A Quadratic Bending Model for Inextensible Surfaces. In *Proc. Eurographics Symp. Geom. Process.* 227–230.

Kiran S Bhat, Christopher D Twigg, Jessica K Hodgins, Pradeep K Khosla, Zoran Popović, and Steven M Seitz. 2003. Estimating cloth simulation parameters from video. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 37–51.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4 (July 2014), 154:1–154:11.

Eddy Boxerman and Uri Ascher. 2004. Decomposing cloth. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 153–161.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Trans. Graph.* 21, 3 (July 2002), 594–603.

R. Bridson, S. Marino, and R. Fedkiw. 2003. Simulation of Clothing with Folds and Wrinkles. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 28–36.

Zhili Chen, Renguo Feng, and Huamin Wang. 2013. Modeling Friction and Air Effects Between Cloth and Deformable Bodies. *ACM Trans. Graph.* 32, 4 (July 2013), 88:1–88:8.

Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. 2014. Yarn-level Simulation of Woven Cloth. *ACM Trans. Graph.* 33, 6 (Nov. 2014), 207:1–207:11.

Gabriel Cirio, Jorge Lopez-Moreno, and Miguel A. Otaduy. 2015. Efficient Simulation of Knitted Cloth Using Persistent Contacts. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 55–61.

Frederic Cordier and Nadia Magnenat-Thalmann. 2002. Real-time animation of dressed virtual humans. In *Computer Graphics Forum*, Vol. 21. 327–335.

Frederic Cordier and Nadia Magnenat-Thalmann. 2005. A Data-Driven Approach for Real-Time Clothes Simulation. In *Computer Graphics Forum*, Vol. 24. 173–183.

Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. 2003. A fast finite element solution for cloth modelling. In *Proc. Pac. Conf. Comput. Graph. Appl.* 244–251.

Ye Fan, Joshua Litven, David I.W. Levin, and Dinesh K. Pai. 2013. Eulerian-on-Lagrangian Simulation. *ACM Trans. Graph.* 32, 3 (July 2013), 22:1–22:9.

Russell Gillette, Craig Peters, Nicholas Vining, Essex Edwards, and Alla Sheffer. 2015. Real-time Dynamic Wrinkling of Coarse Animated Cloth. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 17–26.

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient Simulation of Inextensible Cloth. *ACM Trans. Graph.* 26, 3 (July 2007), 49:1–49:7.

Sunil Hadap, Endre Bangerter, Pascal Volino, and Nadia Magnenat-Thalmann. 1999. Animating Wrinkles on Clothes. In *Proc. Conference on Visualization*. 175–182.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. 2014. Subspace Clothing Simulation Using Adaptive Bases. *ACM Trans. Graph.* 33, 4 (July 2014), 105:1–105:9.

David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. 2009. Asynchronous Contact Mechanics. *ACM Trans. Graph.* 28, 3 (July 2009), 87:1–87:12.

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. 2008. Robust Treatment of Simultaneous Collisions. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 23:1–23:4.

Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2008. Simulating Knitted Cloth at the Yarn Level. *ACM Trans. Graph.* 27, 3 (Aug. 2008), 65:1–65:9.

Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2010. Efficient Yarn-based Cloth with Adaptive Contact Linearization. *ACM Trans. Graph.* 29, 4 (July 2010), 105:1–105:10.

Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. 2011. Physics-inspired Upsampling for Cloth Simulation in Games. *ACM Trans. Graph.* 30, 4 (July 2011), 93:1–93:10.

Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. 2013. Near-exhaustive Precomputation of Secondary Cloth Effects. *ACM Trans. Graph.* 32, 4 (July 2013), 87:1–87:8.

Tae-Yong Kim, Nuttapong Chentanez, and Matthias Müller-Fischer. 2012. Long Range Attachments - a Method to Simulate Inextensible Clothing in Computer Games. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 305–310.

Woojong Koh, Rahul Narain, and James F O'Brien. 2014. View-dependent adaptive cloth simulation. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 159–166.

Cornelius Lanczos. 1986. *The variational principles of mechanics* (4 ed.). Dover.

Duo Li, Shinjiro Sueda, Debanga R. Neog, and Dinesh K. Pai. 2013. Thin Skin Elastodynamics. *ACM Trans. Graph.* 32, 4 (July 2013), 49:1–49:10.

Richard Malgat, Benjamin Gilles, David I. W. Levin, Matthieu Nesme, and François Faure. 2015. Multifarious Hierarchies of Mechanical Models for Artist Assigned Levels-of-detail. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 27–36.

Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A Otaduy, and Steve Marschner. 2012. Data-Driven Estimation of Cloth Simulation Models. In *Computer Graphics Forum*, Vol. 31. 519–528.

Eder Miguel, Rasmus Tamstorf, Derek Bradley, Sara C. Schvartzman, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Steve Marschner, and Miguel A. Otaduy. 2013. Modeling and Estimation of Internal Friction in Cloth. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 212:1–212:10.

Matthias Müller and Nuttapong Chentanez. 2010. Wrinkle Meshes. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 85–92.

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2015. Air Meshes for Robust Collision Handling. *ACM Trans. Graph.* 34, 4 (July 2015), 133:1–133:9.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position Based Dynamics. *J. Vis. Commun. Image Represent.* 18, 2 (April 2007), 109–118.

Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Trans. Graph.* 32, 4 (July 2013), 51:1–51:8.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6 (Nov. 2012), 152:1–152:10.

Simon Pabst, Sybille Krzywinski, Andrea Schenk, and Bernhard Thomaszewski. 2008. Seams and Bending in Cloth Simulation. *VRIPHYS* 382, 1 (2008), 24–41.

Saket Patkar, Ning Jin, and Ronald Fedkiw. 2015. A New Sharp-crease Bending Element for Folding and Wrinkling Surfaces and Volumes. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 7–15.

Tobias Pfaff, Rahul Narain, Juan Miguel de Joya, and James F. O'Brien. 2014. Adaptive Tearing and Cracking of Thin Sheets. *ACM Trans. Graph.* 33, 4 (July 2014), 110:1–110:9.

Xavier Provot. 1996. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *Graphics Interface*. 147–154.

Xavier Provot. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation*. Springer, 177–189.
Olivier Rémillard and Paul G. Kry. 2013. Embedded Thin Shells for Wrinkle Simulation. *ACM Trans. Graph.* 32, 4 (July 2013), 50:1–50:8.
Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. 2010. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-looking Wrinkles. *ACM Trans. Graph.* 29, 6 (Dec. 2010), 157:1–157:8.
Prashant Sachdeva, Shinjiro Sueda, Susanne Bradley, Mikhail Fain, and Dinesh K. Pai. 2015. Biomechanical Simulation and Control of Hands and Tendinous Systems. *ACM Trans. Graph.* 34, 4 (July 2015), 42:1–42:10.
Josep Sarrate, Antonio Huerta, and Jean Donea. 2001. Arbitrary Lagrangian–Eulerian formulation for fluid–rigid body interaction. *Comput. Methods. Appl. Mech. Eng.* 190, 24 (2001), 3171–3188.
Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie C. L. Wang, and Jean-Francis Bloch. 2015. Nonsmooth Developable Geometry for Interactively Animating Paper Crumpling. *ACM Trans. Graph.* 35, 1 (Dec. 2015), 10:1–10:18.
Eftychios Sifakis, Sebastian Marino, and Joseph Teran. 2008. Globally coupled collision handling using volume preserving impulses. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 147–153.
Shinjiro Sueda, Garrett L. Jones, David I. W. Levin, and Dinesh K. Pai. 2011. Large-scale Dynamic Simulation of Highly Constrained Strands. *ACM Trans. Graph.* 30, 4 (July 2011), 39:1–39:10.
Rasmus Tamstorf and Eitan Grinspun. 2013. Discrete Bending Forces and Their Jacobians. *Graph. Models* 75, 6 (Nov. 2013), 362–370.
Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. 2015. Smoothed Aggregation Multigrid for Cloth Simulation. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 245:1–245:13.
Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. In *Computer Graphics*, Vol. 21. 205–214.
Bernhard Thomaszewski, Simon Pabst, and Wolfgang Strasser. 2009. Continuum-based Strain Limiting. *Computer Graphics Forum* 28, 2 (2009), 569–576.
Bernhard Thomaszewski, Markus Wacker, and Wolfgang Straßer. 2006. A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Anim.* 107–116.
Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. 2011. Sensitive Couture for Interactive Garment Modeling and Editing. *ACM Trans. Graph.* 30, 4 (July 2011), 90:1–90:12.
Julien Villard and Houman Borouchaki. 2005. Adaptive meshing for cloth animation. *Engineering with Computers* 20, 4 (2005), 333–341.
Pascal Volino, Martin Courchesne, and Nadia Magnenat Thalmann. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proc. SIGGRAPH 95, Annual Conference Series.* 137–144.
Pascal Volino and Nadia Magnenat-Thalmann. 2000. Implementing Fast Cloth Simulation with Collision Response. In *Computer Graphics International.* 257–266.
Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. 2009. A Simple Approach to Nonlinear Tensile Stiffness for Accurate Cloth Simulation. *ACM Trans. Graph.* 28, 4 (Sept. 2009), 105:1–105:16.
Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F. O'Brien. 2010. Example-based Wrinkle Synthesis for Clothing Animation. *ACM Trans. Graph.* 29, 4 (July 2010), 107:1–107:8.
Huamin Wang, James F. O'Brien, and Ravi Ramamoorthi. 2011. Data-driven Elastic Models for Cloth: Modeling and Measurement. *ACM Trans. Graph.* 30, 4 (July 2011), 71:1–71:12.
Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. 2007. Discrete Quadratic Curvature Energies. *Comput. Aided Geom. Des.* 24, 8-9 (Nov. 2007), 499–518.
Weiwei Xu, Nobuyuki Umentani, Qianwen Chao, Jie Mao, Xiaogang Jin, and Xin Tong. 2014. Sensitivity-optimized Rigging for Example-based Real-time Clothing Synthesis. *ACM Trans. Graph.* 33, 4 (July 2014), 107:1–107:11.

A DERIVATION OF WORLD VELOCITY

Let $\boldsymbol{x}_a$, $\boldsymbol{x}_b$, and $\boldsymbol{x}_c$ be the Lagrangian positions of the three vertices of a triangle, and let $\boldsymbol{X}_a, \boldsymbol{X}_b$, and $\boldsymbol{X}_c$ be their corresponding Eulerian positions. Let $\boldsymbol{X}$ be any material point within this triangle. The world position of this point can be expressed as

$$\begin{aligned}\boldsymbol{x}(\boldsymbol{X}) = &\alpha(\boldsymbol{X}_a, \boldsymbol{X}_b, \boldsymbol{X}_c, \boldsymbol{X})\boldsymbol{x}_a + \\ &\beta(\boldsymbol{X}_a, \boldsymbol{X}_b, \boldsymbol{X}_c, \boldsymbol{X})\boldsymbol{x}_b + \\ &\gamma(\boldsymbol{X}_a, \boldsymbol{X}_b, \boldsymbol{X}_c, \boldsymbol{X})\boldsymbol{x}_c.\end{aligned} \tag{A.1}$$

Here we have made it explicit that the barycentric coordinates, (α, β, γ), are all functions of the query material point, $\boldsymbol{X}$, as well as the Eulerian coordinates of the triangle vertices, $\boldsymbol{X}_a$, $\boldsymbol{X}_b$, and $\boldsymbol{X}_c$. Using the standard expression for converting between barycentric and Cartesian coordinates, we have

$$\begin{pmatrix}\beta \\ \gamma\end{pmatrix} = T^{-1}(\boldsymbol{X} - \boldsymbol{X}_a), \quad T = \begin{pmatrix}\boldsymbol{X}_b - \boldsymbol{X}_a & \boldsymbol{X}_c - \boldsymbol{X}_a\end{pmatrix} \in \mathbb{R}^{2\times 2}. \tag{A.2}$$

Since $\alpha = 1 - \beta - \gamma$, we have

$$\begin{pmatrix}\alpha \\ \beta \\ \gamma\end{pmatrix} = DT^{-1}(\boldsymbol{X} - \boldsymbol{X}_a) + d, \quad D = \begin{pmatrix}-1 & -1 \\ 1 & 0 \\ 0 & 1\end{pmatrix}, \quad d = \begin{pmatrix}1 \\ 0 \\ 0\end{pmatrix}. \tag{A.3}$$

We combine Eq. A.1 and Eq. A.3 to get

$$\begin{aligned}\boldsymbol{x}(\boldsymbol{X}) &= \begin{pmatrix}\boldsymbol{x}_a & \boldsymbol{x}_b & \boldsymbol{x}_c\end{pmatrix}\begin{pmatrix}\alpha \\ \beta \\ \gamma\end{pmatrix} \\ &= \begin{pmatrix}\boldsymbol{x}_a & \boldsymbol{x}_b & \boldsymbol{x}_c\end{pmatrix}\left(DT^{-1}(\boldsymbol{X} - \boldsymbol{X}_a) + d\right).\end{aligned} \tag{A.4}$$

Taking the time derivative, we get

$$\dot{\boldsymbol{x}} = \underbrace{\begin{pmatrix}\dot{\boldsymbol{x}}_a & \dot{\boldsymbol{x}}_b & \dot{\boldsymbol{x}}_c\end{pmatrix}\begin{pmatrix}\alpha \\ \beta \\ \gamma\end{pmatrix}}_{\dot{\boldsymbol{x}}^L} + \underbrace{\begin{pmatrix}\boldsymbol{x}_a & \boldsymbol{x}_b & \boldsymbol{x}_c\end{pmatrix} D\frac{d}{dt}\left\{T^{-1}(\boldsymbol{X} - \boldsymbol{X}_a)\right\}}_{\dot{\boldsymbol{x}}^E}. \tag{A.5}$$

The first term is the Lagrangian contribution, and the second term is the Eulerian contribution. We now focus only on the Eulerian term. First, we note that $T = D_X$, the edge matrix of the Eulerian DOFs from Eq. 3. Similarly, the multiplication of the Lagrangian DOFs by D in the first portion of $\dot{\boldsymbol{x}}^E$ gives D_x, the edge matrix of the Lagrangian DOFs. Taking the time derivative, we have

$$\begin{aligned}\dot{\boldsymbol{x}}^E &= D_x\left(\frac{dD_X^{-1}}{dt}(\boldsymbol{X} - \boldsymbol{X}_a) - D_X^{-1}\dot{\boldsymbol{X}}_a\right) \\ &= D_x\left(-D_X^{-1}\left(\frac{dD_X}{dt}\right)D_X^{-1}(\boldsymbol{X} - \boldsymbol{X}_a) - D_X^{-1}\dot{\boldsymbol{X}}_a\right) \\ &= -D_xD_X^{-1}\left(\left(\frac{dD_X}{dt}\right)\begin{pmatrix}\beta \\ \gamma\end{pmatrix} + \dot{\boldsymbol{X}}_a\right) \\ &= -D_xD_X^{-1}\left(\left(\sum_{i=a}^{c}\frac{\partial D_X}{\partial \boldsymbol{X}_i}\otimes\dot{\boldsymbol{X}}_i\right)\begin{pmatrix}\beta \\ \gamma\end{pmatrix} + \dot{\boldsymbol{X}}_a\right) \\ &= -F\left(\left(\begin{pmatrix}-\dot{\boldsymbol{X}}_a & -\dot{\boldsymbol{X}}_a\end{pmatrix} + \begin{pmatrix}\dot{\boldsymbol{X}}_b & 0\end{pmatrix} + \begin{pmatrix}0 & \dot{\boldsymbol{X}}_c\end{pmatrix}\right)\begin{pmatrix}\beta \\ \gamma\end{pmatrix} + \dot{\boldsymbol{X}}_a\right) \\ &= -F\left(-\beta\dot{\boldsymbol{X}}_a - \gamma\dot{\boldsymbol{X}}_a + \beta\dot{\boldsymbol{X}}_b + \gamma\dot{\boldsymbol{X}}_c + \dot{\boldsymbol{X}}_a\right) \\ &= -F\left(\alpha\dot{\boldsymbol{X}}_a + \beta\dot{\boldsymbol{X}}_b + \gamma\dot{\boldsymbol{X}}_c\right),\end{aligned} \tag{A.6}$$

giving us the final expression for the world velocity:

$$\dot{\boldsymbol{x}} = (\alpha\dot{\boldsymbol{x}}_a + \beta\dot{\boldsymbol{x}}_b + \gamma\dot{\boldsymbol{x}}_c) - F\left(\alpha\dot{\boldsymbol{X}}_a + \beta\dot{\boldsymbol{X}}_b + \gamma\dot{\boldsymbol{X}}_c\right). \tag{A.7}$$