In-Timestep Remeshing for Contacting Elastodynamics

ZACHARY FERGUSON, New York University & Adobe, USA
TESEO SCHNEIDER, University of Victoria, Canada
DANNY M. KAUFMAN*, Adobe, USA
DANIELE PANOZZO*, New York University, USA

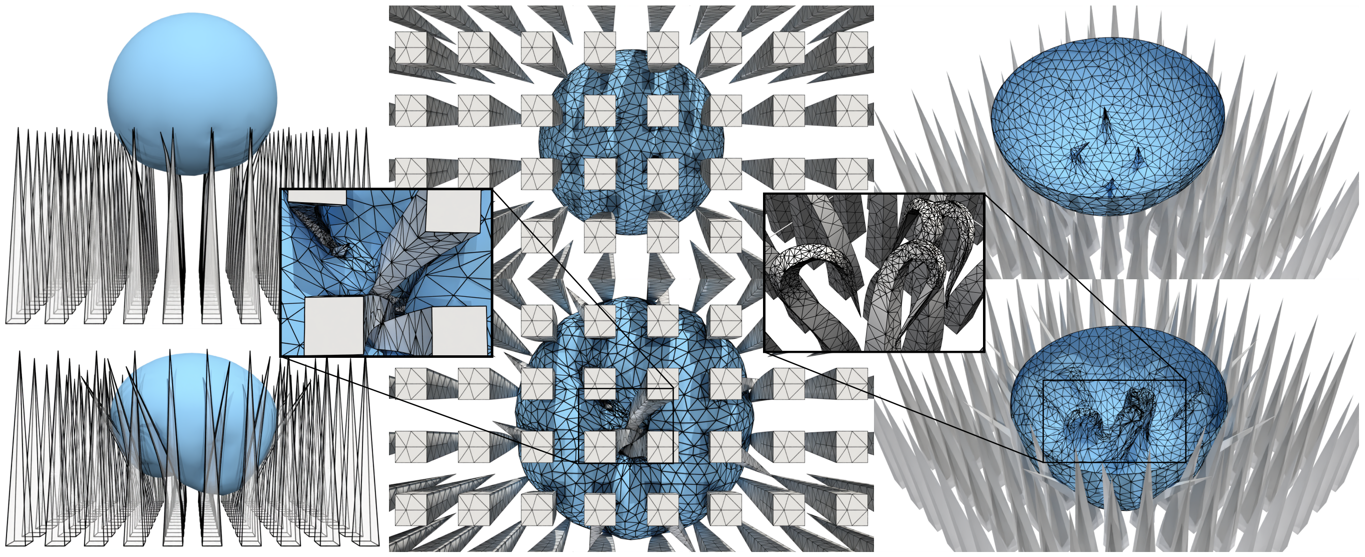

Fig. 1. **Ball on spikes.** In-Timestep Remeshing (ITR) enables physics-aware adaptive refinement and coarsening to robustly capture detailed contact-driven deformations in simulated trajectories. Here we drop a soft (neo-Hookean material, $E = 10^5$ Pa) ball at large timesteps ($h = 0.01$ s) onto very stiff ($E = 10^8$ Pa) sharp spikes. Starting with coarse, unstructured finite-element meshes for all geometries (see Figure 2a) we show here two later steps in the trajectory as the ball initially collides with and then comes to rest on the spikes (top and bottom left respectively). Views from below (middle and middle inset) for each of these steps highlight how our physics-aware remeshing automatically and locally adapts the tetrahedral mesh in time to capture the changing detailed deformations within the material and at contact regions. In a cutaway view (right), we remove the tetrahedral interior elements from the ball, leaving just its bottom surface mesh faces to highlight how ITR tightly conforms, per timestep, without intersection, to the sharp and challenging contacts without over-refining (please compare to the sizing field method in Figure 2b). Correspondingly we cut the ball geometry from the view altogether (right inset) and zoom in on the tightly wound spike geometries that form the severe indentation on the ball, evidencing the accurate solution of the challenging timestep problem resolving forces between highly disparate material stiffnesses.

We propose In-Timestep Remeshing, a fully coupled, adaptive meshing algorithm for contacting elastodynamics where remeshing steps are tightly integrated, implicitly, within the timestep solve. Our algorithm refines and coarsens the domain automatically by measuring physical energy changes within each ongoing timestep solve. This provides consistent, degree-of-freedom-efficient, productive remeshing that, by construction, is physics-aware and so avoids the errors, over-refinements, artifacts, per-example hand-tuning, and instabilities commonly encountered when remeshing with timestepping methods. Our in-timestep computation then ensures that each simulation step's output is both a converged stable solution on the updated mesh and a temporally consistent trajectory with respect to the model and solution of the last timestep. At the same time, the output is guaranteed safe (intersection- and inversion-free) across all operations. We demonstrate applications across a wide range of extreme stress tests with challenging contacts, sharp geometries, extreme compressions, large timesteps, and wide material stiffness ranges – all scenarios well-appreciated to challenge existing remeshing methods.

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

Additional Key Words and Phrases: Adaptive Meshing, Elastodynamics, Variational Contact, Friction

ACM Reference Format:
Zachary Ferguson, Teseo Schneider, Danny M. Kaufman, and Daniele Panozzo. 2023. In-Timestep Remeshing for Contacting Elastodynamics. *ACM Trans.*

*Co-corresponding authors D.K and D.P. jointly conceptualized the study and algorithmic solution.

Authors' addresses: Zachary Ferguson, New York University & Adobe, USA, zfergus@nyu.edu; Teseo Schneider, University of Victoria, Canada, teseo@uvic.ca; Danny M. Kaufman, Adobe, USA, dannykaufman@gmail.com; Daniele Panozzo, New York University, USA, panozzo@nyu.edu.

0730-0301/2023/8-ART1 $15.00
https://doi.org/10.1145/3592428

Graph. 42, 4, Article 1 (August 2023), 15 pages. https://doi.org/10.1145/3592428

1 INTRODUCTION

We propose, In-Timestep Remeshing (ITR), a new algorithm for simulating frictionally contacting elastodynamics where remeshing criteria, remeshing operations, and variable mappings are all tightly coupled implicitly, within the timestep solve. Our algorithm automatically adapts meshing in-timestep to account for time-local conditions of both the internal forces and frictional contacts of a trajectory. At the same time, by careful construction, non-intersection and non-inversion are respected as invariants over each operation within the remeshing, and so across each timestep. This provides consistent improvement across extreme variations in materials, severe boundary conditions, fine surface contact details, large friction, and even under the extreme compressions and tensions regularly imposed by contacting and impacting domains.

Large-deformation elastodynamic simulations often require exceedingly dense spatial discretizations to capture critical and often transient features like shockwaves and indentations. At the same time, meshes dense enough to capture these behaviors can be prohibitively expensive in both runtime and memory for practical applications with real-world examples – especially in 3D. These challenges motivate the application of adaptive meshing (AM) methods that seek to locally introduce and remove simulation degrees of freedom (DOF) on the fly, in order to concentrate them where they are most needed.

Generally, AM for simulating dynamics is currently applied in-between simulation timesteps. This often fits well within optimized physics pipelines in graphics but keeps mesh changes, and the resultant necessary remapping of physical quantities, decoupled from the actual timestep simulation solves. In turn, this decoupling introduces a number of fundamental challenges that we address in this work.

Meshing Criteria. First, the measures evaluated on-mesh that decide where and how to change discretization, can not directly evaluate how remeshing options will impact the solution of the physical problem when decoupled in this way. Applied post-solve, these criteria instead provide approximations, based on the current generated state, at the current, fixed discretization. Indirect proxies are then generally applied, using geometric criteria and/or snapshots of physical quantities to guide the refinement and coarsening, which, in turn, then need to be re-tuned as the specific physical system simulated (e.g., materials, speeds, boundary conditions) change.

Invariants. Second, necessary invariants for accurate, large-deformation contact simulation are often broken in remeshing pipelines, with element inversions and intersections regularly generated. A range of fail safes and stabilizations have been applied to fix these issues *post hoc* [Narain et al. 2013; Spillmann and Teschner 2008]. However, these fixes all have trade-offs: they generally introduce errors, can inject energy (potentially creating instabilities) [Narain et al. 2013], and require per-example tuning even as they work to remove intersections and/or fix elements.

Mapping. Third, physical quantities, e.g., displacements, velocities, and accelerations, must be mapped to new discretizations. Inherently, all such mappings, aside from happy nesting cases, introduce errors. However, the process of alternating timestep solves, meshing, and mapping, additionally introduces inconsistencies between the physical state and the mesh discretization, while mapping operations themselves can also generate intersections and inversions. As we cover in the next sections, this leads to unacceptable artifacts, additional instabilities, and even simulation failures. Prior work in simulating dynamics with adaptive-meshing, in dealing with these issues, often seeks to minimize refinement operations to reduce error [Wicke et al. 2010]. However, this often opposes the original goal of adapting where needed.

Contact. Contact-driven dynamics particularly pose both significant challenges to, and high demand for AM, where large and highly singular contact forces generate significant and localized deformations in simulation meshes. In such cases, the above-covered issues are especially critical to consider as the separation of meshing steps and solves breaks temporal coherence, introduces infeasible states and unnecessarily perturbs system energies (with attendant numerical artifacts and jittering), and so often undoes much of the immediate benefit of improved accuracy and quality targeted by AM operations in the first place. Likewise, existing contact-aware AM methods, applying solely geometric criteria significantly over-refine boundaries (see Figure 5), in many cases again directly opposing the original intent of AM.

In-Timestep AM. We address the above challenges with, to our knowledge, a first fully coupled AM method for contacting elastodynamics with meshing criteria, operations, and mappings, tightly coupled within each timestep solve. To do so we apply the recently proposed, Incremental Potential Contact (IPC) model [Li et al. 2020] which provides a convergent [Li et al. 2023] and smooth model for frictionally contacting solids Applying the IPC model, we construct In-Timestep Remeshing where meshing criteria have access to the current, ongoing, nonlinear timestep solve's merit function. With this framework, we can apply efficient "micro" simulations per mesh operation, and so make physics-informed and invariant-safe decisions on how to update the discretization.

Contributions. ITR thus refines and coarsens by measuring the change in improvement within each ongoing timestep solve and so avoids recourse to geometric meshing criteria that are physics-oblivious and require per-example tuning.

To build ITR our technical contributions include:

- a "safe" constrained L^2-projection method for variable mapping that minimizes mapping error while preserving invariants by ensuring a globally injective mapping;
- a consistent, smooth remeshing criteria function for frictionally contacting elastodynamics built upon the IPC model; and
- a refinement and coarsening algorithm with provably safe operations, operation filtering heuristics for limiting per-step cost while ensuring solution improvement, and local nonlinear analysis leading to a final, convergent timestep solution on each step's new mesh.

Runtime Efficiency. When compared to uniform mesh refinement, ITR judiciously adds and removes DOF, reducing linear solve times in the inner loop of the nonlinear timestepping algorithm with a DOF improvement ranging from 2.6 to 185× less DOF per example with corresponding 2.7 to 1,444× linear solve speedups. However, additional computation is applied to select where and when the spatial discretization is modified. The interplay between resultant time-savings in linear solves and this overhead varies significantly depending on the scene (and how suitable a naive refinement is). Scenes requiring localized refinement (e.g., Figure 1) are up to 3.3× faster with our implementation of ITR, while others (e.g., Figure 6) can be up to 9.6× slower. We provide a detailed analysis of this trade-off and discuss its long-term implications for this technology in Section 4.3.

We demonstrate the effectiveness of our approach across a wide range of challenging 3D (and 2D) examples, where we highlight the benefits of physics-aware AM. While simple methods are desirable, remeshing necessarily comes with the cost of complex implementation: we release a modular, open-source implementation of our methods at polyfem.github.io to enable replicability and future application.

2 RELATED WORK

Meshes are ubiquitous in graphics and there are a wide range of algorithms and applications that create and/or modify them. We focus here specifically on related work on unstructured meshes and the application of mesh modifications for elastodynamic simulation, both with and without frictional contact. For a broad overview of adaptive methods in graphics covering a wide range of physical problems, models, and structured discretizations, please refer to Manteaux et al. [2017]. Hu et al. [2018] similarly review dedicated meshing algorithms, and Mitchell and McClain [2014] cover methods combining mesh modifications with *basis refinement* (p and hp-refinement), which we do not consider in this work.

Adaptive remeshing also plays a critical role in modeling fracture and cutting [Chentanez et al. 2009; Hahn and Wojtan 2015; Koschier et al. 2015; Manteaux et al. 2015; O'Brien et al. 2002; O'Brien and Hodgins 1999; Pfaff et al. 2014] as well as in surface tracking methods which employ complex and robust remeshing operations to explicitly track the movement of colliding and merging boundaries [Brochu and Bridson 2009; Da et al. 2014; Jiang et al. 2017; Klingner et al. 2006; Menon et al. 2015; Misztal et al. 2014; Misztal and Bærentzen 2012; Müller et al. 2015; Stein et al. 2004; Wojtan et al. 2009]. Here we focus solely on elastodynamic simulation without fracture and look to extensions in these areas as exciting potential future directions.

Changing a physical model's spatial discretization during elastodynamic simulation requires four high-level algorithmic components:

(1) *Criteria:* where to change the discretization and, when doing so, where to increase or decrease the number of DOF;
(2) *Operations:* which operations are applied to change the discretization;

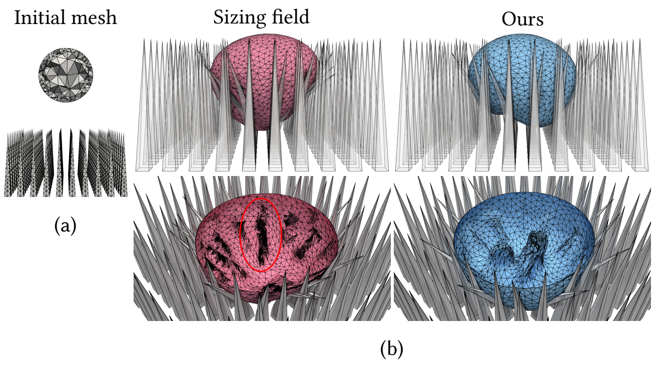

Fig. 2. **Sizing field comparison.** (a) The initial conditions and mesh used for the "ball on spikes" simulations in Figures 1 and 2b. (b) A comparison of our algorithm (right) and results of applying a contact-aware sizing field-based adaptive meshing criteria [Li et al. 2018; Narain et al. 2012; Wicke et al. 2010] (left) for in-timestep simulation. We show a cutaway view (bottom row) where we have clipped the geometry to see the inside of the sphere's surface. While the sizing field result refines around the contacts, it severely over-refines right away (circled in red) and so fails to capture intricate interactions. In comparison, our method adaptively updates while tracking both contact and internal forces and so locally refines to capture the spikes pushing into the ball (see Figure 1 for a closer view of our results).

(3) *Mapping:* once a discretization is changed, how physical quantities are mapped from the prior discretization to the new one; and
(4) *Solution Schedule:* how and when these mapped quantities are applied to update the physical model's solution.

In the following, we next categorize and consider related works with respect to their treatment of these four core components.

2.1 Criteria

Geometry. Starting from the seminal work of Hutchinson et al. [1996] for mass-spring systems, a popular way of guiding simulation mesh adaptation is to rely on the geometry of the discretization, either in rest configuration [Bargteil et al. 2007], deformed configuration [Dunyach et al. 2013], or both, enabling the use of a snapshot of strains or stresses [Bargteil et al. 2007; Debunne et al. 2001; Spillmann and Teschner 2008; Wicke et al. 2010]. Similar criteria have been proposed for shells [Li and Volkov 2005; Narain et al. 2013, 2012; Simnett et al. 2009; Villard and Borouchaki 2005], where additional considerations for the complex in-plane and bending behaviors of thin materials play an important role. Additionally, and interestingly, user-dependent geometric criteria such as camera view [Koh et al. 2015] can be considered for refinement. These measures are then primarily proxies for the variations in physical energy in the system, and for the quality of the underlying discretization to represent it. They are, however, approximations based solely on the current rest and deformed configurations at the current, fixed discretization.

Contact. Contacts pose both significant challenges to, and high demand for, adaptive remeshing. Contact forces generate large, yet localized, deformations in many simulation meshes and regularly introduce highly singular strains on boundaries for which it is often desirable to improve resolution. Geometric *proximity* criteria are often applied to help select regions for remeshing on simulation mesh boundaries where two or more surfaces are geometrically close [Bender and Deul 2013; Erhart et al. 2006; Simnett et al. 2009]. Proximity alone is often insufficient and so is sometimes augmented by temporal continuity conditions of the detected collisions [Spillmann and Teschner 2008], and even higher-order approximations that consider a contact region's curvature via contact tangents across mesh faces [Li et al. 2018; Narain et al. 2013, 2012; Pfaff et al. 2014]. While often effective, these measures do not account for the actual contacting geometry, force balance between contacts and the elastic materials (e.g., considering whether materials involved are equally stiff and so less likely to deform and require adaptation), contact force magnitudes, nor the frictional forces involved. With purely geometric analysis these underlying material and configurational aspects are unaccounted for and so opportunities for necessary refinement and useful coarsening on the contact regions are missed – leading to significant over-refinement or under-refinement in many cases; see Section 4.1 and Figures 2b and 5 for examples and evaluation.

Elastic Energy. Rather than apply geometric proxies, a number of recent works focus on applying criteria that measure a model's elastic energy as a criterion in assessing the effectiveness of remeshing [Demkowicz 2006; Mitchell and McClain 2014]. Most closely related to our approach Mosler and Ortiz [2007] propose elastic- and incremental-plastic energy decrease as criteria for small remeshing problems in elastostatics and plasticity, but are limited to solely refinement operations, and do not address contact, friction, nor dynamics.

2.2 Operations

Global. Global methods [Jiang et al. 2017; Klingner et al. 2006; Skouras et al. 2014; Stein et al. 2004], create a new mesh for every timestep. Often this is applied via an external meshing tool and so gives the advantage of reducing the implementation effort that would otherwise be required for tighter integration. However, in building a new mesh from scratch, opportunities for problem-aware and ideally smaller updates are lost while such large global changes in the simulation mesh, necessarily incur larger errors in mapping; see Section 2.3 below.

Local. Local methods applied in simulation [Li et al. 2018; Narain et al. 2013, 2012; Spillmann and Teschner 2008] utilize a sequence of local remeshing operations (splits, collapses, swaps/flips) to modify the mesh according to the criteria applied (Section 2.1). While applied locally these operations can still cause trouble by creating intersections that must be prevented [Brochu and Bridson 2009] and inversions [Wicke et al. 2010], while also potentially injecting error by introducing instabilities if not resolved carefully (Section 2.4). We apply local mesh operations in concert with invariant checks and post-operation energy evaluations to guarantee effective (error-decreasing) and safe (invariant-preserving) mesh adaptations.

Basis and r-Refinement. An alternative to explicit changes in the mesh is to adaptively refine the basis, either via h-refinement within-element [Grinspun et al. 2002] or via p-refinement [Mitchell and McClain 2014]. While adaptive, these methods are not designed to deal with large deformations as they cannot change the shape of the elements (the mesh is fixed). Complementary adaptivity is also provided by r-adaptive or "moving-mesh" methods [Budd et al. 2009] which update the nodal locations in the deforming model's rest mesh but not the topology. While effective in capturing localized dynamics behavior [Zielonka et al. 2008], on its own r-adaptivity is not suited for dynamic contact problems, which generally require concentrated refinement in highly local and often rapidly changing regions.

2.3 Mapping

Closest Point. An efficient approach to transfer vertex-based quantities between two meshes in close spatial proximity is to transfer the attributes from a vertex/quadrature point of one mesh to its closest neighbor on the other [Molinari and Ortiz 2002]. This approach introduces large errors when the meshes have elements of different sizes, and usually requires post-stabilization techniques (Section 2.4) to avoid simulation artifacts, especially in the presence of stiff materials and contact.

Interpolation. A more accurate method with a bit larger computational overhead is to interpolate via the finite element basis – when linear elements are applied, this is equivalent to the barycentric coordinate interpolation commonly applied in graphics [Spillmann and Teschner 2008; Wicke et al. 2010]. Despite higher accuracy, significant errors still accumulate and post-stabilization techniques remain necessary [Narain et al. 2013; Spillmann and Teschner 2008].

L^2 Projection. Given the above issues, a natural strategy is to compute a mapping that minimizes error [Léger et al. 2014; Vavourakis et al. 2013]. The L^2 projection finds the representation of the function in the finite element space of the target mesh that is a least-squares fit of the function in the finite element space of the source mesh [Léger et al. 2014], and so minimizes the residual of the mapping. Considerably more expensive and challenging to implement than the above alternatives, this projection is commonly applied in scientific computing and mechanical engineering.

We advocate, to our knowledge, for the first time in the graphics community, the L^2 projection for adaptive mesh refinement, as it is robust to both varying mesh densities and low-quality elements. However, despite these important properties, the L^2 projection, on its own, remains insufficient for large-deformation dynamics as it can not ensure necessary invariants in elastodynamics are preserved. In particular, the projection can create intersections and element inversions. In Section 3.5, we provide a brief overview of the L^2 projection and then propose our extension to obtain an invariant-preserving, error-minimizing mapping.

2.4 Solution Schedule

For elastodynamic simulation, a fundamental question is how to integrate remeshing and mapping variables into each timestep's solution of the physical model.

Interleaving. The standard strategy is to decouple timestepping from remeshing, generally by interleaving timestep solve, remeshing, and mapping. This leaves the remeshing criteria to the mercy of *post hoc* quantities, while after remeshing, the newly mapped variables are finalized as the updated state for the timestep [Narain et al. 2012; Wicke et al. 2010]. Except for local remeshing operations that create nested spaces, the above-covered mappings (Section 2.3) all necessarily introduce errors in the projected quantities – meaning the newly mapped solution is inaccurate and inconsistent with the underlying mesh it is defined on and so introduces artifacts including instabilities and jittering [Narain et al. 2013]. Moreover, the updated solution can introduce intersections and inversions, generated by the prior mapping. Earlier works, recognizing these issues, do apply geometric corrections for intersections [Narain et al. 2012] but, at the same time, generally strive to minimize the overall number of remeshing operations to reduce total error [Narain et al. 2012; Wicke et al. 2010].

Post-Stabilization. Post-stabilization methods, recognizing the instability and jittering introduced by direct mapping of timestepped variables to the new mesh, introduce an additional step, after mapping, to improve stability (although not accuracy). Narain et al. [2013], apply a nonlinear least-squares solve to perturb mapped positions to find a more stable configuration, while Spillmann and Teschner [2008] apply a similar strategy with additional collision response phases to also correct for intersections. These methods, with proper parameter tuning, can be effective at removing visual artifacts, such as jittering, but they also introduce significant errors in the physical model, as they can apply arbitrary perturbations to a solution that already contains errors.

2.5 IPC and In-Timestep Remeshing

The above analysis leads us to the conclusion that for simulating elastodynamics, the remeshing criteria, remeshing operations, and variable mappings, should all be tightly coupled, and so integrated together within *each* timestep solve. The next question is how. Mosler and Ortiz's [2007] work on simulating elastostatics using the elasticity potential is our starting point for contacting elastodynamics. We begin by applying the recently proposed, IPC model [Li et al. 2023, 2020] which provides a convergent and *smooth* model for frictionally contacting solids. In turn, because IPC contact forces are smooth and the IPC timestep update is variational, this allows us to formulate, per timestep, a spatially smooth merit function as our meshing criteria. This merit function includes elasticity, contact, and friction, and its decrease guarantees solution improvement. We then carefully design our solver to refine, coarsen and safely L^2-project, within each timestep solve, to maintain consistent updates, while ensuring that the final output of each timestep is an accurate, intersection-free, and inversion-free solution progressing the simulation dynamics forward in time.

Hierarchical Methods. Hierarchical methods provide solver strategies, complementary to AM methods, that can be applied to improve timestep solves. These methods (e.g., [Hormann et al. 1998; Zhang et al. 2022]) apply a hierarchy of pre-determined resolutions to better compute a solution for a final (pre-specified) and generally uniform, high-resolution target mesh.

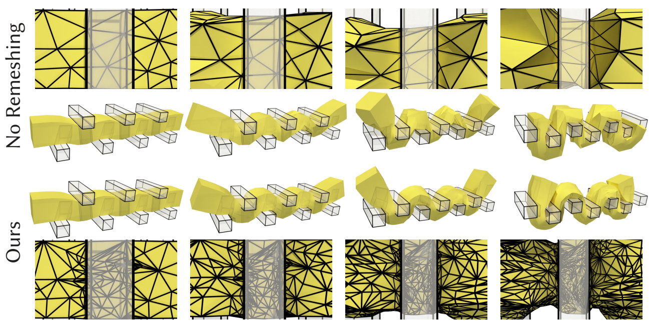

Fig. 3. **Masticator.** A challenging 3D compression example, simulated without refinement (top) and with (bottom) our algorithm, starting from the same initial mesh. The insets highlight how our method is able to capture the sharp contact features and buckling under compression by increasing mesh resolution. Without remeshing, these details are lost, resulting in a different deformation.

In contrast, AM methods (including ITR) locally adapt solution meshes to apply detailed resolution where it can be better used. In future work, it should be an interesting extension to consider the application of hierarchical methods within ITR to obtain faster nonlinear solves. For this, the most closely related approach to ITR in the hierarchical literature, is the recent work of Zhang et al. [2022] who build a hierarchical solver for IPC. They propose a Euclidean projection to find non-intersecting geometries nearest to possibly-intersecting, Loop-subdivision-upsampled targets, by applying barrier-enforced, continuous collision detection (CCD)-filtering to the direct path from a "safe", midpoint-upsampled triangle mesh, to a target. Here we construct a complementary, error-minimizing, L2-projection, for tetrahedral meshes, constructed by constrained quadratic energy minimization, supplemented with CCD-filtered collision barriers, suitable for refinement and coarsening operations.

3 IN-TIMESTEP REMESHING

3.1 Spatially Continuous Setting

We consider the solution of simplicial simulation meshes (triangles in 2D, tetrahedra in 3D) undergoing large-deformation elastodynamics with frictional contact. Before discretizing to a spatial mesh, we first begin by discretizing *in time*: we construct the solution of each timestep's problem in semi-discrete optimization form,

$$x^{t+1} = \operatorname*{argmin}_{x} E_t(x) \tag{1}$$

with a spatially continuous Incremental Potential,

$$\begin{aligned} E_t(x) = &\int_\Omega \frac{\rho}{2} \left\| x(X) - \tilde{x}^t(X) \right\|^2 \, \mathrm{d}V \\ &+ \alpha h^2 \int_\Omega \Psi\big(x(X)\big) - x(X)^\top f(X) \, \mathrm{d}V \\ &+ \alpha h^2 \int_{\partial\Omega} B\big(x(X)\big) + D\big(x(X)\big) \, \mathrm{d}A. \end{aligned} \tag{2}$$

Here Ψ is a hyperelastic deformation-energy density (e.g. neo-Hookean), f encodes the sum of body forces and (when ranging over boundary regions) any applied reactions, and B and D are the spatially-continuous analogs of the IPC energies [Li et al. 2023] for, respectively, contact barrier and friction pseudo-potential. In turn, the choice of predictor position, $\tilde{x}^t$ (an explicit function of prior deformation, velocity, and possibly acceleration fields: $x^t, x^{t-1}, \ldots, v^t, v^{t-1}, \ldots, a^t, a^{t-1}, \ldots$), scaling term $\alpha \in \mathbb{R}^+$, and an explicit update equation for velocity (and acceleration as needed) from optimal solution x^{t+1}, jointly define the specific choice of numerical time-integration method. Here, in the main text, for simplicity, we will keep in mind implicit Euler with

$$\tilde{x}^t = x^t + hv^t,\ v^{t+1} = \frac{1}{h}(x^{t+1} - x^t),\ \alpha = 1, \tag{3}$$

while similarly, a wide range of additional time integration methods are directly covered[1].

3.2 Solution Quality per Timestep

In this continuum form, the optimization timestep solve in Equation (2) highlights an important deciding feature: when we are allowed to range over the space of all valid deformations x, a deformation giving the (locally) smaller value of E_t is the better solution to the timestep. Looking ahead to our next step of spatial discretization, this provides a simple, physics-focused metric for ranking finite-element meshes in a solution space, that is custom-suited to each *timestep*. Of course, a corollary is that this energy decrease is always "easily" obtained via uniform refinement to finer and finer meshes but this also comes with the associated cost of more, and generally too much, computation. Instead, here we focus on applying this metric to locally adapt our mesh in regions of high value. To do so we focus our adaptivity on this actual, temporally local, measured change in the timestep's solution quality itself, rather than on intermediate proxies via mesh qualities or physical properties.

3.3 Spatial Discretization

We apply piecewise-linear discretization of Equation (2) on meshes $\mathcal{T}$ with discrete fields defined, per triangulation/tetrahedralization, at the n vertices of the mesh in 3D (respectively 2D) space and stored in vectors $x, v, a \in \mathbb{R}^{3n}$ (respectively $\mathbb{R}^{2n}$).

Each of the spatially discrete energy terms in our incremental potential are now expressed as weighted sums of energy functions over mesh element stencils, s (tetrahedral, triangle, edge, point or pairings thereof depending on energy and dimension) in $\mathcal{T}$,

$$\sum_{s \in \mathcal{T}} w_s W_s\left(x\right),$$

where $w_s > 0$ is the volume, area or length-weighted scaling of the rest shape element s, and W_s is the respective energy density function of each potential restricted to this element's stencil.

[1]For example, other time-integration methods we applied are Implicit Newmark, with

$$\alpha = 1/4, \tilde{x}^t = x^t + hv + h^2/4a^t, v^{t+1} = 2/h\left(x^{t+1} - x^t\right) - v^t, \text{and}$$
$$a^{t+1} = 2/h\left(v^{t+1} - v^t\right) - a^t,$$

and second-order backward differentiation formula (BDF2), with

$$\alpha = 4/9, \tilde{x}^t = \tfrac{1}{3}(4x^t - x^{t-1}) + \tfrac{2h}{9}(4v^t - v^{t-1}), a^{t+1} = \tfrac{4h^2}{9}(x^{t+1} - \tilde{x}^t), \text{and}$$
$$v^{t+1} = \tfrac{1}{3}(4v^t - v^{t-1}) + \tfrac{2h}{3}a^{t+1}.$$

Small changes by additional terms in the arguments of the energy functions extend the range of our application even further to a yet wider range of numerical time integration methods without loss of generality [Li et al. 2023].

For a fixed mesh $\mathcal{T}$, the timestep solution is then a local minimizer of a fully discrete Incremental Potential, per timestep

$$\begin{aligned} E_t(x, \mathcal{T}) =& E(x, \mathcal{T}, \tilde{x}^t) \\ =& \frac{1}{2}(x - \tilde{x}^t)^\top M_{\mathcal{T}}(x - \tilde{x}^t) \\ &+ \alpha h^2\big(\Psi_{\mathcal{T}}(x) + B_{\mathcal{T}}(x) + D_{\mathcal{T}}(x) - x^\top f^t\big), \end{aligned} \tag{4}$$

where $M_{\mathcal{T}}$ is the mesh's consistent mass matrix, and $\Psi_{\mathcal{T}}, B_{\mathcal{T}}$, and $D_{\mathcal{T}}$ are the total resultant energy potentials generated, respectively, by the aforementioned discretizations of the corresponding deformation, contact barrier, and friction energies on $\mathcal{T}$ [Li et al. 2023].

3.4 Timestepping Framework and Invariants

We advance our simulation domain through time using an incrementally updating triangulation of the domain, $\mathcal{T}(t)$, with deformations $x(t)$, velocities $v(t)$, and rest positions, $\bar{x}(t)$ defined at $\mathcal{T}(t)$'s vertices. Input for the solve of each timestep optimization is then: $\bar{x}^t, x^t, v^t$, and current applied forces (body and external), f^t, defined on mesh $\mathcal{T}^t$ with the *deformed* mesh $(x^t, \mathcal{T}^t)$ giving a penetration- and inversion-free configuration.

In turn output for each of our timestep solves is then a new mesh $\mathcal{T}^{t+1}$ and updated fields, $\bar{x}^{t+1}, x^{t+1}, v^{t+1}$ that maintain the invariants of non-intersection and non-inversion at end state, while accurately satisfying the numerical time-integration model by minimizing the incremental potential with $\left\|\nabla_x E_t(x, \mathcal{T}^{t+1})\right\| \leq \epsilon_d$.

At the same time, as we cover in detail below, to better resolve dynamics, each of our timestep solves also incrementally updates the simulation mesh $\mathcal{T}$, as a "configurational" degree of freedom with mesh-refinement to lower the total value of the incremental potential solution in Equation (2) measured by

$$m_t(\mathcal{T}) = \min_x E_t(x, \mathcal{T}), \tag{5}$$

and similarly coarsening where this does not significantly increase this same value.

Maintaining Invariants. We apply Newton iterations to minimize $E_t(x, \mathcal{T})$ when holding the mesh fixed. Preserving invariants for these steps we follow the IPC method's filtered line-search step [Li et al. 2020] which applies CCD and inversion-checking to descent steps. This ensures that all applied displacements for position updates to x ensure both safety and energy decrease towards convergence. In our setting with remeshing, this is not enough – all operations during each timestep computation, including remeshing, must maintain non-intersection and non-inversion at every update.

3.5 Safe Projections Between Spaces

Each remeshing operation, i, applied during a timestep solve changes the mesh, $\mathcal{T}^i \rightarrow \mathcal{T}^{i+1}$. This means that all quantities, $(x^t, v^t, f^t, \bar{x}, \ldots)$ defined in the prior mesh must, of course, be mapped, or *projected*, to the new one.

When simulating dynamics these quantities are generally transferred in-between timestep solves (see Section 2.3); that is given $\bar{x}, x^t, v^t$, and $\mathcal{T}^1$ we would first solve for a new timestep solution, x^{t+1}, v^{t+1}, then update the mesh to a new one $\mathcal{T}^2$. Then, only after remeshing, x^{t+1}, v^{t+1}, are projected to the new mesh. Unfortunately, this staggered process means that $x^{t+1} \neq \operatorname{argmin}_x E_t(x, \mathcal{T}^2)$ and so is not a solution to the timestep problem on the current mesh. This inconsistency between solution space and deformation then commonly generates instabilities and jittering artifacts, especially when dealing with stiffer materials and collisions (see e.g., Narain et al. [2013] and their supplemental video).

Another fundamental challenge for remapping variables is the actual definition of the projection operator itself. As alluded to above, changing the mesh also changes the underlying function space of the model. This is why inconsistencies from staggering the timestep solves and projections can generate such significant errors when timestepping.

A cheap, popular, and perhaps simplest projection strategy is closest-point sampling where we assign new nodal values from the closest node in the prior mesh. While tempting, this "projection" introduces large artifacts and instabilities when elements' sizes differ [Vavourakis et al. 2013], e.g., under refinement, where many new nodes' values are often assigned from the same source node in the prior mesh. A popular alternative is to apply interpolation from the finite-element basis – barycentric interpolation in our linear-element setting. This generally gives better results than closest-point sampling, but still introduces large projection errors, again leading to artifacts [Léger et al. 2014; Vavourakis et al. 2013; Wicke et al. 2010].

We instead begin with the L^2 projection that *minimizes* the two-norm error residual when we map from starting to target finite-element space [Léger et al. 2014]. Consider again updating from $\mathcal{T}^1$ with function space V^1 and basis $\{\varphi_i^1 \mid 1 \le i \le n\}$ to $\mathcal{T}^2$ with corresponding function space V^2 and basis $\{\varphi_i^2 \mid 1 \le i \le m\}$. We now define a least-squares projection operator

$$\mathcal{P}\colon V^1 \to V^2, \tag{6}$$

so that for functions $f^1 \in V^1$ their projection $f^2 = \mathcal{P}(f^1) \in V^2$ minimizes the L^2 residual $\frac{1}{2}\left\|f^1 - f^2\right\|^2$. Optimality conditions minimizing this residual [Léger et al. 2014] give the projection of a quantity u, defined on the vertices of $\mathcal{T}^1$ (e.g., the coefficient vector of f^1), to the vertices of $\mathcal{T}^2$ as

$$\widetilde{M}_{\mathcal{T}^2}^{-1} A_{\mathcal{T}^2}^{\mathcal{T}^1} u, \tag{7}$$

where $\widetilde{M}_{\mathcal{T}^2} \in \mathbb{R}^{m\times m}$ is the density-normalized mass matrix on $\mathcal{T}^2$ and $A_{\mathcal{T}^2}^{\mathcal{T}^1} \in \mathbb{R}^{m\times n}$ is a transfer matrix between bases so that $a_{ij} = \int_\Omega \varphi_i^2 \varphi_j^1 \, dV$. The two bases are then defined on two different meshes and so we use arrangement via PolyClipper [Powell 2021] to compute the quadrature points necessary for the integral [Krause and Zulian 2016].

Although not, to our knowledge, previously applied in graphics, this L^2-projection has long been appreciated in mechanics applications for its better preservation of quantities [Léger et al. 2014] due to minimized error. However, while well-projecting *unconstrained* quantities the L^2 projection (and all others) are oblivious to our invariants. Projections can and will create both element inversions and intersections, meaning we can not apply the L^2 projection operator as-is.

To make the L^2 projection safe we return to the variational picture and now rebuild a constrained least-squares residual minimization, *subject to non-intersection and non-inversion* constraints, that safely

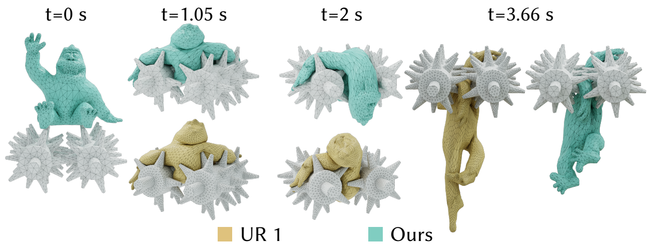

Fig. 4. **Gorilla rollers.** A very soft gorilla model ($E = 2 \times 10^4$ Pa) is dropped on a pair of stiff rotating elastic rollers ($E = 2 \times 10^8$ Pa) with softer spikes ($E = 2 \times 10^7$ Pa). As the gorilla impacts the spikes, the mesh is refined to account both for the large elastic forces in thin features and for the rapidly changing contact forces. Our method adapts to the different material stiffness, by refining the softer gorilla in the necessary regions of contact, much less for the stiffer spikes, and leaves the even stiffer roller unadapted. The dynamics for the single-level uniformly-refined (UR) solution (UR 1) is comparable up to $t = 2$ s where the spike is (unlike the adapted mesh solution) is unable to push into the gorilla's left shoulder.

projects quantities from an old mesh $\mathcal{T}^{\text{old}}$ to a new one, $\mathcal{T}$,

$$P_{\mathcal{T}}(u) = \operatorname*{argmin}_v \tfrac{1}{2} v^\top \widetilde{M}_{\mathcal{T}} v - v^\top A_{\mathcal{T}}^{\mathcal{T}^{\text{old}}} u + B_{\mathcal{T}}(x) + I_{\mathcal{T}}(x). \tag{8}$$

The first two terms form the least squares condition, $B_{\mathcal{T}}$ is our discretized IPC contact barrier defined on the new mesh, and $I_{\mathcal{T}}$ discretizes a new barrier we propose to enforce non-inversion during projection without biasing the solution with elastic material behavior,

$$I_{\mathcal{T}}(x) = \sum_{t\in\mathcal{T}} w_t c_t(x, \hat{v}), \tag{9}$$

where, the function c_t returns a log barrier on the volumes $v_t(x)$ of tetrahedra t, that is smoothly activated when volume falls below $\hat{v}$,

$$c_t(x,\hat{v}) = \begin{cases} -\kappa_v \left(\frac{v_t(x)}{\hat{v}} - 1\right)^2 \ln\left(\frac{v_t(x)}{\hat{v}}\right), & 0 < v_t(x) < \hat{v} \\ 0 & v_t(x) \ge \hat{v}. \end{cases} \tag{10}$$

We use $\hat{v} = 10^{-12}\,\text{m}^3$ and $\kappa_v = 0.1E$ (same as contact barrier stiffness) throughout where E is the material's Young's modulus. To apply each constrained projection we first safely initialize our displacement variables on the new mesh via linear interpolation and then directly reuse our same line-search-filtered Newton method to solve Equation (8) and so minimize the L^2-residual while ensuring safe new variables on the updated mesh.

For all of our ITR phases, detailed in the next two sections, all prior timestep quantities $(\cdot^t)$ must be projected to ensure consistency. However, in our setting, we are able to take advantage of a simple optimization: during refinement (only), barycentric interpolation is equivalent to our L^2 projection and so can be safely and cheaply applied rather than Equation (8) for all our edge-split operations.

3.6 Remeshing with Local Operations

Changing the *geometry* (i.e., vertex rest-positions in our setting) of a mesh is attractive for mesh adaptation as it leads to continuous changes in the underlying finite element space, and so is amenable to gradient-based optimization of functionals depending on them. However, such r-adaptive-type updates are insufficient to capture the large, often transient, and highly localized deformations we capture in accurate elastodynamic modeling. Instead, we often require increasing (and decreasing) the number of DOF and so the number of vertices in our meshes. However, in changing the *connectivity* of a mesh we obtain discontinuous changes in our finite-element space. In turn, this will drive nonsmooth changes in our meshing criteria functional in Equation (2) and so here we will apply a discrete optimization strategy.

We consider two types of operations that are applicable to both triangle and tetrahedral meshes: 1) an edge split, which splits every triangle/tetrahedra touching an edge into two while inserting a vertex, and 2) its inverse, an edge collapse. These operations are discrete in nature, but depend on both discrete and continuous parameters: a split operation is applied to a discrete edge, but the position of the newly inserted vertex is controlled by two or three continuous coordinates.

Due to the discrete nature of the problem, it is not practical to seek an optimal sequence of operations minimizing Equation (5) (presuming a fixed sequence length or a targeted given tolerance). We instead apply a greedy block-coordinate descent strategy: we test a set of potential operations and pick those that provide maximal local improvement in the energy for the inserted DOF. We first discuss how we evaluate the effect of a single operation, and then how to greedily select a sequence of operations reducing Equation (5).

Effect of an individual operation. To evaluate the effect of an operation on Equation (5) a naive approach would be to perform the mesh modification, project quantities (Section 2.3), ensure that the invariants are still valid, and minimize Equation (5) globally. However, this is computationally prohibitive: inspired by approaches used for *a posteriori* error estimators [Mitchell 1991; Schmidt and Siebert 2000], we perform local solves in the neighborhood of the mesh modification. This enables a sound approximation of each operation's impact, under the assumption that this effect decays as we move from the operation's stencil. By changing the neighborhood's size, we trade accuracy of our estimator with computational cost.

Scheduling. While we can not tractably obtain globally optimal sequences for meshes, we could potentially find locally optimal solutions by always continually selecting splits that satisfy a sufficient amount of energy decrease (i.e., a minimal necessary reduction) of our energy until no more remain. However, as we scale to larger (and 3D) meshes, this approach is no longer practical either. Detailed below we thus introduce a culling method that preemptively discards candidate split operations that are not likely to lead to large energy reductions (and correspondingly discards candidate edge collapses that are likely to lead to energy increases). We do this by filtering based on the elastic and contact energies per mesh element, with greater local energy concentration indicating a higher likelihood (although not guaranteed) of energy reduction benefit by splitting. Note that our filtering heuristic is applied solely to *cull* likely ineffective operations – our criteria for acceptance remains unchanged by it. We detail our filtering method in the next section.

Implementation. Implementing this discrete optimization algorithm is challenging, especially for tetrahedral meshes, as we need a mechanism to *preview* each connectivity change, extract its patch, and minimize Equation (5). If the operation is invalid, or else does not satisfy our criteria, changes to the connectivity and to its associated fields defined on our mesh need to be rolled back. This is significantly challenging to implement via low-level libraries, e.g., CGAL, libigl, or OpenMesh. We opt to implement our remeshing with declarative specification in Jiang et al. [2022], which allows us to explicitly work on the mesh before and after the operation, and directly supports invariant checks and rollbacks.

3.7 In-Timestep Remeshing Algorithm

We provide a high-level overview of our method in Algorithm 1.

Initial Timestep Solution. Give a current solution state x^t, v^t from the last timestep solve[2] at time t, our ITR first computes a new predictor timestep solution x' by minimizing Equation (4) on the current mesh $\mathcal{T}_t$ (Line 3).

Refinement. Using the new solution x', we sort every edge e_i according to its elastic energy $\Psi_{\mathcal{T}_t}(e_i)$ (area-weighting all adjacent cells) to form list E_Ψ, and create list E_B by sorting e_i according to its contact energy $B_{\mathcal{T}_t}(e_i)$ (averaged over two adjacent faces in 3D) (Line 6). We then select the top ϵ_S% of both lists to form the filtered set S of edges as candidates for splitting operations (Line 7).

We then proceed to the Split procedure (Line 8). The Split procedure takes the set of candidate operations S, and for each operation performs the split (Line 25), obtaining a new mesh $\mathcal{T}'_t$, updates the variable on the mesh by linear interpolation along the split edge[3] (Line 26), which for each split is equivalent to a zero-error L^2 projection, and then performs a small local solve (Line 27). See the next section below for details on the *Local Solve*. The split operation is accepted if we obtain sufficient decrease, $\delta E = E_{t+1}(x_i, \mathcal{T}_i) - E_{t+1}(x_p, \mathcal{T}_p) > \delta_s$, and the newly created edges are applied to update the queue (Line 31). Otherwise, if the operation is rejected for providing insufficient improvement, we undo the split.

Local Solve. Local solves applied in both the *Split* procedure above and the *Collapse* procedure below follow the same procedure. A timestep re-solve is performed in a local patch by minimizing Equation (4) on the current mesh, but now fixing *all nodes* in the system except for DOF in a local patch with a size that is the maximum between the 2-ring of the edge and 1% of the domain's volume, and using x' as a safe and "near-to-solution" warm start. A first i iterations are run ($i = 4$ for contacting patches and 1 otherwise) and then checked to see if it reaches respectively, sufficient decrease for a split (see above) or small (by $|\delta_c|$) acceptable increase for a collapse (see below). If the remeshing criteria *is not reached* the operation is abandoned (as covered) as the Newton decrement shows no progress. Otherwise, if the criteria are met and we will be accepting the operation we continue the local-patch solve to convergence

[2]For clarity in pseudocode and discussion we do not track the update of a^t, a^{t+1} nor x^{t-1} here. Treatment for acceleration terms, when time-integration methods are applied that use them, follow identically to v^t, v^{t+1} throughout, similarly treatment of x^{t-1} follows identically to x^t.

[3]In practice, we use a simple averaging of endpoint values.

Algorithm 1 Overview of our in-timestep remeshing algorithm.

```
1:  procedure InTimestepRemeshing($x_t, v_t, \mathcal{T}_t$)
2:      // Initial Timestep
3:      $x' \leftarrow \mathrm{argmin}_x E_t(x, \mathcal{T}_t)$
4:
5:      // Refinement
6:      $E_\Psi \leftarrow$ Sort($\{\Psi_{\mathcal{T}_t}(e_i)\}$), $E_B \leftarrow$ Sort($\{B_{\mathcal{T}_t}(e_i)\}$)
7:      $S \leftarrow E_\Psi > \epsilon_S \cup E_B > \epsilon_S$,
8:      $\mathcal{T}'_t, x'_{t+1}, x'_t, v'_t \leftarrow$ Split($S, \mathcal{T}_t, x', x_t, v_t$)
9:      $x_t, v_t \leftarrow x'_t, v'_t$
10:
11:     // Coarsening
12:     $C \leftarrow E_\Psi < \epsilon_C \cap E_B < \epsilon_C$,
13:     $\mathcal{T}_{t+1}, x'_{t+1}, x'_t, v'_t \leftarrow$ Collapse($C, \mathcal{T}'_t, x'_{t+1}, x'_t, v'_t$)
14:     $x_t, v_t \leftarrow$ SafeProject($\mathcal{T}_t, \mathcal{T}_{t+1}, x_t, v_t$)
15:
16:     // Global Solve
17:     $x_{t+1} \leftarrow \mathrm{argmin}_x E_t(x, \mathcal{T}_{t+1})$
18:     return $\mathcal{T}_{t+1}, x_{t+1}, x_t, v_t$
19: end procedure
20:
21: procedure Split($S, \mathcal{T}, x_{t+1}, x_t, v_t$)
22:     $Q \leftarrow$ BuildPriority($S$)
23:     while $Q \neq \emptyset$ do
24:         $e \leftarrow$ Pop($Q$)
25:         $\mathcal{T}' \leftarrow$ SplitEdge($e, \mathcal{T}$)
26:         $x'_{t+1}, x'_t, v'_t \leftarrow$ Interpolate($x_{t+1}, x_t, v_t, \mathcal{T}, \mathcal{T}'$)
27:         $x'_{t+1} \leftarrow$ LocalSolve($x'_{t+1}, \mathcal{T}'$)
28:         if $E_t(x_{t+1}, \mathcal{T}) - E_t(x'_{t+1}, \mathcal{T}') > \delta_s$ then
29:             $\mathcal{T} \leftarrow \mathcal{T}'$
30:             $x_{t+1} \leftarrow x'_{t+1}$, $x_t \leftarrow x'_t$, $v_t \leftarrow v'_t$
31:             $Q \leftarrow$ UpdateQueue($Q, \mathcal{T}$)
32:         end if
33:     end while
34:     return $\mathcal{T}, x_{t+1}, x_t, v_t$
35: end procedure
36:
37: procedure Collapse($C, \mathcal{T}, x_{t+1}, x_t, v_t$)
38:     $Q \leftarrow$ BuildPriority($C$)
39:     while $Q \neq \emptyset$ do
40:         $e \leftarrow$ Pop($Q$)
41:         $\mathcal{T}' \leftarrow$ CollapseEdge($e, \mathcal{T}$)
42:         $x'_{t+1}, x'_t, v'_t \leftarrow$ Interpolate($x_{t+1}, x_t, v_t, \mathcal{T}, \mathcal{T}'$)
43:         if InvariantCheck($x'_t, x'_{t+1}, \mathcal{T}'$) then
44:             $x'_{t+1} \leftarrow$ LocalSolve($x'_{t+1}, \mathcal{T}'$)
45:             if $E_t(x_{t+1}, \mathcal{T}) - E_t(x'_{t+1}, \mathcal{T}') > \delta_c$ then
46:                 $\mathcal{T} \leftarrow \mathcal{T}'$
47:                 $x_{t+1} \leftarrow x'_{t+1}$, $x_t \leftarrow x'_t$, $v_t \leftarrow v'_t$
48:                 $Q \leftarrow$ UpdateQueue($Q, \mathcal{T}_t$)
49:             end if
50:         end if
51:     end while
52:     return $\mathcal{T}, x_{t+1}, x_t, v_t$
53: end procedure
```

(same termination tolerance as the global solve) ensuring that downstream operations (and the final solve) all start from well-resolved regions.

Coarsening. Next, we then select the bottom ϵ_C% from E_Ψ and E_B to form the set C of candidate edges for potential *collapse* operations (Line 12) and attempt to collapse them (Line 13).

As in *Split*, the *Collapse* procedure takes a set of candidate operations C, and for each operation performs the collapse (Line 41), obtaining a new mesh $\mathcal{T}'_t$. However, differently from *Split*, the collapse operation *does not* create a nested space, and so interpolation will introduce mapping-errors. In turn, these errors can occasionally break our invariants. To avoid this problem, we locally perform an interpolation, and then explicitly check if the invariants are violated (Line 43). If they are violated, we reject the operation, otherwise we proceed similarly to check that the *Split* operation *does not increase* system energy by more than a small, prescribed tolerance via a $\delta_c \leq 0$ and otherwise follow as in the *Collapse* procedure.

As in the interpolation applied in our *Split* operations, we interpolate the endpoints of the collapsed edge to determine each new vertex's attributes. We collapse boundary edges if and only if neighboring faces are coplanar in order to preserve the mesh's rest shape. For the same reason, when an edge has a single vertex on the boundary, we collapse it to the boundary endpoint. For all other edges, we average the endpoints.

After all collapse operations are complete, unlike after splits, we now require a L^2-projection of prior displacements and velocities on $\mathcal{T}_{t+1}$ by means of the safe L^2 projection (Line 14) described in Section 3.5, using our interpolated quantities as safe initialization.

Global Solve. Finally, warm-starting with the latest solution estimate x'_{t+1}, we perform a final re-solve of Equation (4) on the full domain using the finalized new mesh $\mathcal{T}_{t+1}$ and then explicitly update velocity to v_{t+1}. As we have been incrementally updating (effectively relaxing) the solution throughout this process this final solve is efficient (the number of iterations for convergence is low) as the majority of the effort has been performed in both the initial and intermediary solves during remeshing.

4 EVALUATION

Our algorithm is implemented in C++, using Eigen [Guennebaud et al. 2010] for basic linear-algebra, PolyFEM [Schneider et al. 2019] for finite element (FE) system construction, IPC Toolkit [Ferguson et al. 2020] for evaluating IPC potentials and collision detection, Wildmeshing-toolkit [Jiang et al. 2022] for mesh data structures and editing, Pardiso [Alappat et al. 2020; Bollhöfer et al. 2019, 2020] for the large linear systems in our global Newton solves, and Eigen's dense Cholesky decomposition ($LL^\top$) for the small linear systems in our local Newton solves. All experiments are run on a cluster node with an Intel Cascade Lake Platinum 8268 processor limited to 16 threads. Our reference implementation, used to generate all results, will be released as an open-source project. Please see our supplemental video for result animations and Table 3 for parameters used.

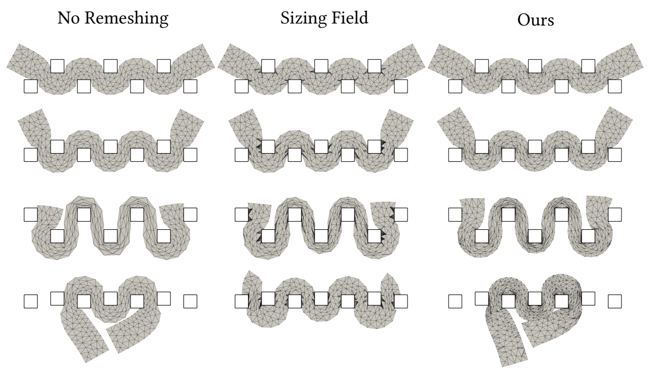

Fig. 5. **2D Masticator.** Simulation of the deformation of a 2D bar deformed by a set of squares. the same mesh is used for the simulation without remeshing (left column), remeshed using a sizing field based on [Li et al. 2018; Narain et al. 2012; Wicke et al. 2010] within our in-timestep framework (middle column), and with ITR ("Ours", right column). ITR produces a more detailed simulation compared to the run without remeshing, adding DOF to accurately capture the sharp contact with the cubes and the large deformation of the bar. The method based on the sizing field overrefines the contact regions (and does not refine elsewhere) and leads to a different "snagged" final configuration.

4.1 Comparisons

To our knowledge, our ITR algorithm is the only AM method that can ensure the preservation of IPC invariants and so can be combined with the IPC contact model. This is because mappings and contact failsafes applied in previous works can and will fail with intersections and downstream failures in challenging contacting scenarios like those we test here (Section 2). In order to compare with prior methods on these challenging scenarios we focus on comparing *meshing criteria* in a comparable side-by-side setting allowing all methods to utilize within-timestep simulation and IPC solves.

To robustly process contacts and implicitly solve dynamics with IPC we replace our physics-aware meshing criteria within ITR with Wicke et al.'s [2010] sizing field, based on the deformation gradient [Wicke et al. 2010, Equation (7)], for internal deformation criteria, and on the most recent, state-of-the-art contact sizing criteria proposed by Li et al. [2018]. We compare our energy-based acceptance criteria with this sizing field[4]. To do so we replace our criteria in our implementation with the above sizing field and accept edge-split and edge-collapses following the scheduling of Narain et al. [2012] (same as Li et al. [2018]).

We instrument two side-by-side comparison examples: one in 2D (*Masticator*) and the other in 3D (*Ball-on-Spikes*). Initially, we observe that the contact-based sizing field leads to runaway endless refinement on contacting surfaces – rapidly leading to intractable simulations. On closer inspection, we see that division by contact distance in the denominator of Li et al.'s [2018] sizing tensors is the source: here accurate IPC contact-processing allows for exceedingly close compliance between surfaces. To enable the contact sizing field strategy to progress we add a limit to the method restricting edge lengths to 0.01.

For our 2D example, we see in Figure 5 that both our algorithm and the sizing field method improve on the simulation of the original unrefined mesh (left column). However, the sizing field greedily refines in contact regions with large numbers of unnecessary faces that can significantly slow simulation and lead to overly compliant surfaces locally, that "snag" on boundaries. Please see our supplemental video for detailed trajectories of all three simulations.

Similar results bear out for our 3D test in Figure 2b. Here we again see that our remeshing criterion automatically adapts to both the contact geometries, the relative material stiffnesses of the domains, and the force balance between the coiled spikes and the dropped ball – leading to the resolution and local mesh adaptation necessary to capture all these details. In contrast, we again see that the sizing field rapidly over-refines for the first initial contacts, leading to highly meshed, but minor, side indentations for the first initial collisions with the spikes, but entirely misses the later spike protrusions and compressed coiling as it does not account for the physical solution and relative forces (compare with Figure 1). Please also see our supplemental video for more details and a comparison with the simulation of the unrefined starting mesh.

4.2 Results

Sharp Contact. In Figure 3, we reproduce the *Masticator* example of Wicke et al. [2010] with a large timestep ($h = 0.05$ s) to stress-test a deformable bar compressed by a set of rigid boxes. Our algorithm quickly captures the sharp contact interfaces upon collision, followed by more refinement to allow compliance along the block, curvature on top, and initially symmetric bulging of the bar out-of-plane, followed by the start of buckling. In contrast, without refinement, the simulated bar's initial mesh does not have sufficient DOF to capture contact and compliance – these behaviors are missed and instead we obtain a jagged and twisted deformation.

Large Deformation with Self-Contact. In Figure 6, a stiff bar ($E = 10^7$ Pa) is anchored on both sides and twisted by rotating its top. In the close-ups we see how prior to contact our algorithm progressively refines the tetrahedral domain as more winding introduces greater curvature and more stress. As winding continues, the simulation adapts to provide even twisting along bar faces and edges until buckling. Upon buckling, we observe in the zoom-in of Figure 6 how our simulation of the bar adapts the mesh to capture the collapse, fold-in, and exceedingly tight frictional contact of its faces (e.g., the middle red face). In contrast, simulating directly (unrefined) with the initial bar misses these details and leads to large deformation errors even before the onset of buckling.

Complex geometry and material-awareness. Varying surface complexity and material stiffness across domains are likewise simultaneously resolved by our algorithm. Here a stiff roller is scripted to rotate, with slightly softer spikes and much softer dropped gorilla geometry in Figure 4. On contact, we see the gorilla geometry increasingly refines around the impact site with the bar (which refines less due to a stiffer reaction) and then coarsens as it rebounds.

[4]Note that we do not apply the additional deformation sizing field criteria from [Li et al. 2018; Narain et al. 2012] as those are customized for shell models – for this, we take our deformation sizing component instead from the volumetric work of Wicke et al. [2010].

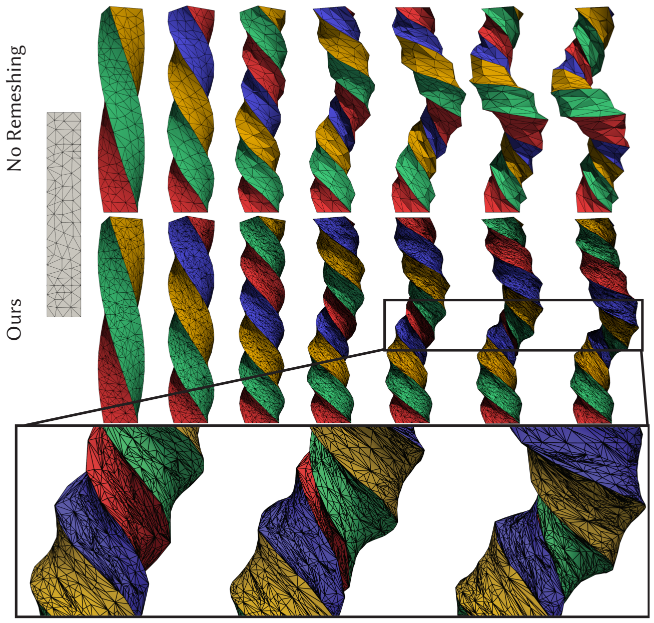

Fig. 6. **Bar-twist.** Starting from the same coarse geometry (shown in grey) the resulting deformed mesh is very different without (top) and with (middle) ITR. Our algorithm adaptively adds (and removes) DOF in the mesh to better resolve elastodynamics. As our adaptive simulation progresses we move from regular twisting to buckling. Bottom inset: during buckling our physics-aware remeshing allows the face to collapse (see the center red face) and fold in on itself with tightly resolved contact.

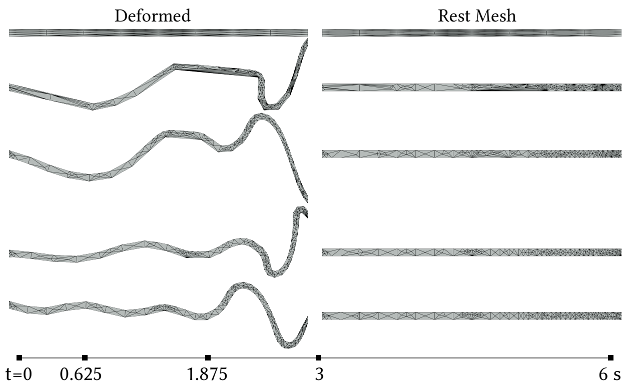

Fig. 7. **Elastic wave.** Simulation of a driven periodic wave motion. Starting from a coarse rectangular mesh (top), ITR progressively adapts the simulation mesh with increasing corresponding resolution from left to right and local adaptations in appropriate regions to capture the steady wave dynamics (bottom).

High-Speed Impact. While above we stress-test ITR with a number of large timestep examples, for many phenomena we may wish to capture detailed deformation occurring at much finer time scales. In Figure 8, we model the high-speed impact test from Li et al. [2020] with ITR. A soft ($E = 10^6$ Pa) ball is fired at a wall obstacle with high velocity ($v_0 = 67\,\mathrm{m/s}$) with a timestep of $h = 2 \times 10^{-5}$ s. Here ITR begins with a much coarser (17× fewer tetrahedra) initial mesh. Then, on impact, ITR *automatically* begins refining the mesh to capture both the rapidly changing contact interface on the surface and the internal propagation of shock waves across the material. It then *coarsens* the mesh as the shockwave passes through, with light, secondary refinement and coarsening applications capturing the subsequent oscillations in free-flight. Please see our supplemental video for the resulting dynamics of the simulation and the changing mesh supporting it.

Dynamic Wave Propagation. In many cases, emergent behavior in a system's dynamics would best define a suitable mesh choice – but this is hard to predict without a higher-resolution simulation result to guide us in the first place. In Figure 7, we fix the left side of a coarsely triangulated beam and then drive its other end with periodic vertical oscillations. Appropriately modeled, this driven system should lead to a steady state of attenuating waves damping as they traverse the bar from right to left. Over multiple timesteps, we see that ITR progressively adapts the simulation mesh with increasing corresponding resolution left to right and local adaptations in appropriate regions to smoothly capture the steady wave dynamics.

Energy Effectiveness of Remeshing. We instrument the above ball-impact example to study the effectiveness of our ITR (Figure 9). Across the entire simulation, we compute the energy decrease per timestep of the total incremental potential energy obtained by remeshing from the beginning of the timestep solve (prior to our algorithm initializing the remeshing operation: Algorithm 1 Line 3) to the final solution output (Algorithm 1 Line 17) on the timestep's adapted mesh: $\Delta E = E(x_{t+1}, \mathcal{T}_{t+1}) - E(x', \mathcal{T}_t)$. In Figure 9 we see that as we refine the mesh (noticeably just around the first contact, marker (a)) our method dramatically improves the energy, while during coarsening (e.g., after complete separation in (d)) the energy does not increase, despite the removal of DOF. Looking more closely at the trends we also see proportionately more energy decrease when more refinement operations are performed demonstrating the effectiveness of our criteria's selection and timing of operations.

Stability. As covered in Section 2, a fundamental challenge in AM methods for dynamics, especially during coarsening, is stability. Each ITR timestep solve in all the above examples is solved to convergence on the timestep's final output mesh with the prior state safely L2-projected to it. We observe that qualitatively (see our supplemental video), all the trajectories generated by ITR remain stable, and so free of jittering and instability artifacts, e.g. as demonstrated in prior methods by Narain et al. [2013].

4.3 Performance and Resolution

For non-adapted (fixed mesh) timestep solves of IPC there are three primary sources of computational cost per Newton iterate: (1) evaluation of the energy potential gradients and Hessians, and their assembly to a global linear system, (2) the linear solve of each such system to compute a descent direction, and (3) line search along this direction. Both (1) and (3) involve the evaluation of potential-energy

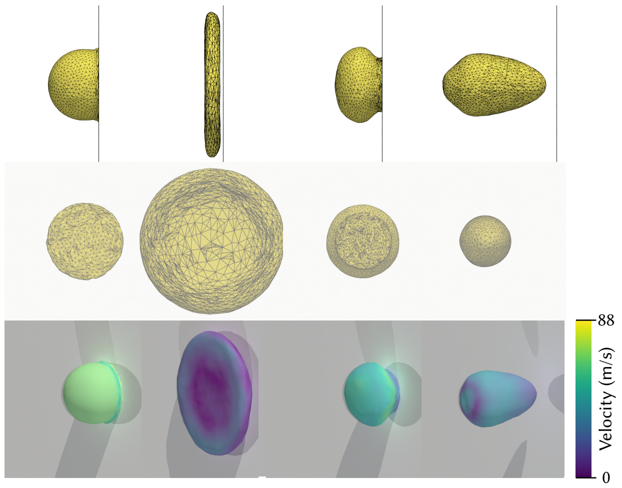

Fig. 8. **Impacting ball.** We replicate the high-speed impact test from [Li et al. 2020] where a soft ($E = 10^6$ Pa) ball is fired at a static wall with a high velocity ($v_0 = 67$ m/s). Beginning with a much coarser initial mesh, ITR's adaptive remeshing determines that refinement only begins during initial collision (left column). As the ball bounces away from the wall, ITR then begins removing DOF, which were earlier added to capture the contact dynamics and are now unnecessary. The bottom row shows the velocity magnitude throughout this process.

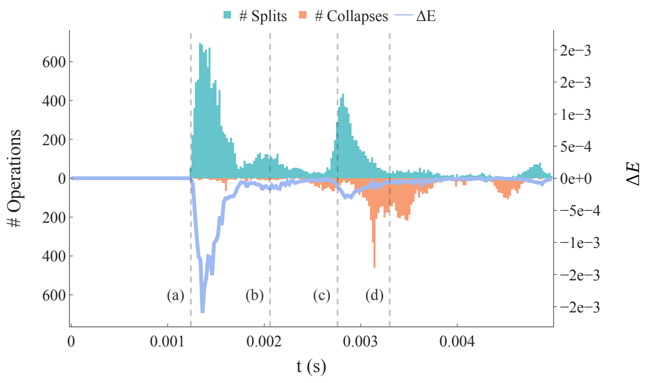

Fig. 9. For each timestep of the impacting ball simulation (Figure 8), we plot the number of splits (green bars), the number of collapses (orange bars), and the change in energy (blue line) from the initial solve of the timestep (prior to remeshing operations) via minimization of E to the final solution of the timestep on the final updated mesh. Key times in the simulation are indicated by virtual lines: (a) first contact, (b) maximal compression, (c) rebound of the material as peels away, and (d) complete separation. The plot shows significant improvements (decrease) in the energy as we apply splitting operations, and at the same time, the coarsening operations do not negatively affect the energy, while increasing efficiency.

stencils and collision detection. As such, they generally dominate costs only for exceedingly small systems since, with modern acceleration strategies and easy parallelization, they generally scale close to linear in the number of elements. On the other hand, the large, ill-scaled, sparse linear system solves per Newton iteration in (2) dominate the costs for all practical examples as they require direct solvers [Li et al. 2020] with poor parallel scaling of a memory bound problem [Lipton et al. 1979].

By adapting the mesh ITR significantly lowers DOF count for high-quality simulation output (see Table 1) and so reduces the size of the largest (super-linear cost) solver bottleneck: linear system sizes. At the same time, ITR introduces a new potentially large (albeit linear and currently unoptimized) overhead cost per timestep to evaluate the suitability of each mesh-adaptation proposal.

To evaluate the current runtime performance of ITR with respect to these computational costs, we compare ITR with successive uniform refinements [Ong 1994] (splitting along the first diagonal) in two inter-related analyses. In the first, we measure the *wall-clock* time used by each implementation, including current (unoptimized) costs for ITR's remeshing overhead. Here we report both running time and memory consumption, noting that there is only one value for memory as linear solves are the bottleneck for memory usage. In the second, we consider an *ideal* analysis; we keep in mind that linear solver technology is an advanced, exceptionally well-studied domain with little expectation of significant improvement, while costs for ITR mesh adaptivity are currently linear and not yet addressed in our ITR implementation with significant optimization nor even low-hanging opportunities for parallelization. For the latter, we focus on the DOF difference (for comparable quality output) and the corresponding difference in global system linear solve times for the IPC timestep solver.

Statistics for these comparisons are reported in Tables 1 and 2. We begin by visually identifying, per benchmark example, the artifact-free baseline UR simulation with qualitatively comparable results to our ITR simulation. As a concrete example, consider the ball on-spike scene, where we observe that simulations with both one and two levels of refinement exhibit significant snagging and severe element distortion; please see Figure 10 for examples. On the other hand, for the gorilla roller scene, two-levels of uniform refinement are sufficient to remove most artifacts and obtain qualitatively similar deformation and contact compliance to our ITR result, please see Figure 11.

In summary, we see ITR's DOF improvement ranging from 2.6 to 185× less DOF per example with corresponding 2.7 to 1,444× linear solve speedups. At the same time, the impact of our initial, unoptimized implementation of our remeshing procedures on wall-clock time varies significantly across examples, ranging from 3.3× speedup for the complex ball on spikes scene to 9.6× slowdown for the much simpler bar-twist scene.

Opportunities for wall-clock performance improvement. We identify four high-impact and immediate directions for future extensions that we believe will likely provide significant improvement in remeshing costs (and so runtimes) for ITR: (1) the most immediate and low-hanging opportunity is the development of parallel and distributed versions of ITR; (2) similarly low-hanging is the application of custom collision-detection that is spatially localized to leverage the small local support that our individual mesh operations evaluate in our local-solve updates (currently this is still applied

Table 1. The average running time per timestep, peak memory, and the number of DOF for an unrefined mesh (UR 0), three levels of uniform refinement (UR 1–3), and our method. We bold the values corresponding to the lowest resolution showing comparable and artifact-free results to ITR.

Scene	Average running time per step (s)					Peak memory (GiB)					Number of DOF				
	UR 0	UR 1	UR 2	UR 3	Ours	UR 0	UR 1	UR 2	UR 3	Ours	UR 0	UR 1	UR 2	UR 3	Ours avg (max)
Ball on spikes (Figure 1)	20.1	418.1	8,371.6	**40,648.1**	**12,316.6**	0.9	3.7	24.3	**151.5**	**2.0**	27k	144k	900k	**6M**	**43k (55k)**
Masticator (Figure 3)	70.8	480.7	**9,781.7**	16,775.8	**8,230.8**	0.9	0.5	**2.7**	16.2	**4.1**	1k	9k	**63k**	476k	**16k (57k)**
Gorilla rollers (Figure 4)	15.2	162.0	**3,317.3**	23,372.5	**1,077.8**	1.0	3.8	**24.7**	182.6	**7.7**	18k	115k	**800k**	5M	**22k (27k)**
Bar-twist (Figure 6)	0.2	1.5	**232.5**	2,733.6	**2,234.9**	0.1	0.4	**3.1**	21.1	**3.7**	1k	9k	**63k**	476k	**24k (78k)**
Impacting ball (Figure 8)	0.3	5.9	**115.8**	8,564.0	**960.8**	0.2	1.1	**8.3**	71.8	**2.9**	5k	34k	**248k**	1M	**50k (61k)**

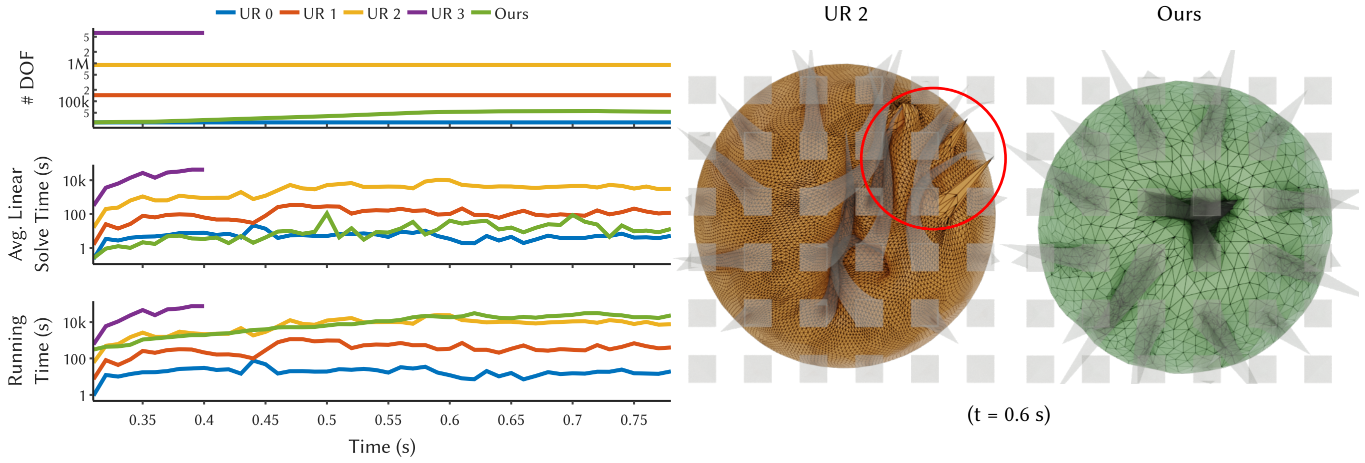

Fig. 10. **Ball on spikes uniform comparison.** Here we plot a detailed view of the performance of the ball on spikes simulation (Figure 1). We compare an unrefined mesh (UR 0), three levels of uniform refinement (UR 1–3), and our method. Circled in the rendering on the right, it is clear that UR 2 was insufficient in capturing the local deformations and stretching caused by the spike tips.

Table 2. Average linear solver running time.

Scene	UR 0	UR 1	UR 2	UR 3	Ours
Ball on spikes (Figure 1)	0.08	0.81	9.88	201.77	0.25
Masticator (Figure 3)	0.01	0.06	0.78	9.39	0.22
Gorilla rollers (Figure 4)	0.12	1.23	16.82	245.44	0.17
Bar-twist (Figure 6)	0.00	0.05	0.57	8.77	0.21
Impacting ball (Figure 8)	0.02	0.28	3.60	126.68	0.37

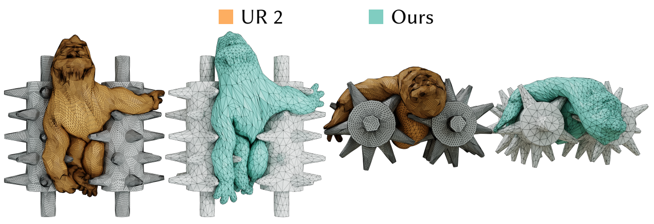

Fig. 11. **Gorilla rollers uniform comparison** demonstrates results of the gorilla roller simulation with two levels of uniform refinement (UR 2) and ITR ("Ours") at the halfway point of the simulation.

globally), (3) exploiting both temporal- and spatial-coherence during collision-detection and culling, and (4) exploration of higher-order bases and geometry to further reduce DOF count.

5 DISCUSSION

We have proposed ITR, a first fully-coupled adaptive-remeshing algorithm for implicit timestepping elastodynamics with frictional contact via a spatially continuous incremental potential merit function. To do so ITR ensures non-penetration and non-inversion throughout all operations in both remeshing and solving. In turn, it applies robust physics-aware remeshing to generate stable and accurate trajectories with low DOF counts. Simulated geometries conform well to necessary contacting interfaces and deformations with parsimonious refinement where new DOF are needed to improve the solution, and effective coarsening where they are not.

5.1 Limitations and Future Work

Along with the opportunities for improved remeshing operation performance discussed above in Section 4.3 we see a number of additional avenues for fruitful improvements and extensions.

Currently, we empirically demonstrate improved solution behavior on a wide range of challenging examples. However, an important next step, which we do not address here is a formal convergence study of our refinement. We provide a preview of such a study in Figure 12, where we consider a cantilever convergence test (see e.g., Pelteret [2016]). As can be seen, ITR's convergence is currently highly dependent on the initial discretization. At least in part, this dependence appears closely related to controlling for mesh *quality*.

Table 3. IPC simulation and ITR parameters. For each example, we report the timestep size (h), density (ρ), Young's modulus (E), Poisson ratio (ν), barrier activation distance ($\hat{d}$), barrier stiffness (κ), coefficient of friction (μ), friction accuracy parameter (ϵ_v), and max friction iteration setting. We also report the split and collapse acceptance tolerances ($\delta_{S,C}$) and the culling thresholds ($\epsilon_{S,C}$). For all examples, we use a Newton convergence criteria of $\|\Delta x\|/h \leq 10^{-3}$ m/s, implicit Euler time integration, and a neo-Hookean material.

Scene	h (s)	ρ (kg/m^3), E (Pa), ν	$\hat{d}$ (m), κ (Pa)	μ, ϵ_v (m/s), friction iters.	δ_s (J), δ_c (J)	ϵ_S, ϵ_C
Ball on spikes (Figure 1)	0.01	2000/1100, 1e5/1e8, 0.4	1e-3, 6e4	0.1, 1e-3, 1	1e-5, -1e-8	0.95, 0.01
2D Masticator (Figure 5)	0.05	1e3, 1e4, 0.4	1e-3, 1e3	0.1, 1e-3, 1000	1e-3, -1e-8	0.95, 0.01
3D Masticator (Figure 3)	0.05	1e3, 1e4, 0.4	1e-3, 1e3	0.1, 1e-3, 1000	1e-5, -1e-8	0.95, 0.01
Gorilla rollers (Figure 4)	0.01	1e3, 5e4/2e7/2e8, 0.3	1e-3, 3e5	0.5, 4e-3, 1	1e-5, -1e-8	0.95, 0.01
Bar-twist (Figure 6)	0.01	1e3, 1e7, 0.4	1e-3, 1e6	-	1e-3, -1e-8	0.95, 0.01
Elastic wave (Figure 7)	0.025	1340, 1e4, 0.495	1e-3, 1e3	-	5e-5, -1e-8	0.85, 0.01
Impacting ball (Figure 8)	2e-5	1150, 1e6, 0.45	6.9e-5, 1e5	-	1e-14, -1e-16	0.95, 0.01
Cantilever (Figure 12)	0.1	1e6, 1.1e9, 0.3	1e-3, 1.1e8	-	*,-1e-13	0.6, 0.4

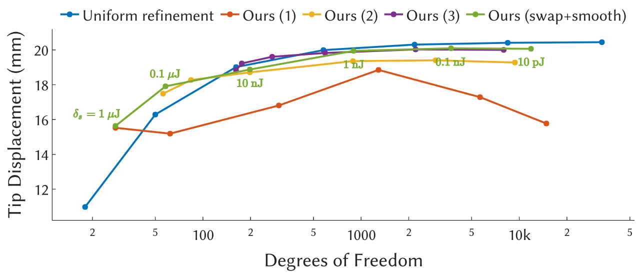

Fig. 12. **Cantilever convergence.** Using a cantilever example (see e.g., Pelteret [2016]), we examine the convergence behavior of ITR with varying refinement acceptance tolerances δ_s from 1 μJ to 10 pJ. We observe that the accuracy of our method is largely dependent on the initial discretization: Ours (1–3) start from 1 to 3 levels of initial refinement, respectively. As a proof-of-concept, we also test a preliminary extension of ITR that additionally utilizes edge-swapping and vertex smoothing, starting from the same mesh as Ours (1). Here we see that these operations are important to "breakaway" from the initial discretization.

We observe that while split and collapse operations are effective for adaptive updates, they are insufficient to preserve mesh quality and so limit the convergence and range of refinement that ITR can currently apply. The inclusion of edge/face flips/swaps will be a simple and direct improvement that naturally fits within the ITR framework, as will be explicit optimization for mesh quality [Wicke et al. 2010]. As a proof-of-concept investigation, in the above cantilever experiment, we have updated ITR operations to additionally include a preliminary version of edge flips and vertex smoothing. As we see in Figure 12 this improves ITR's convergence.

Additional extensions of ITR's adaptivity to also include r-refinement should also be valuable. Likewise, while we focus here solely on volumetric elastodynamics, ITR should usefully extend to codimensional models for shell and rod simulations and even alternative contact models.

We hope that this work and its reference implementation will encourage further research on the application of adaptive unstructured remeshing. As simulation methods advance and problem complexities grow, it becomes all the more important to judiciously apply computation where it can be most effective.

ACKNOWLEDGMENTS

This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. This work was also partially supported by the NSF CAREER award under Grant No. 1652515, the NSF grants OAC-1835712, OIA-1937043, CHS-1908767, CHS-1901091, NSERC DGECR-2021-00461 and RGPIN 2021-03707, a Sloan Fellowship, and a gift from Advanced Micro Devices, Inc.

REFERENCES

Christie Alappat, Achim Basermann, Alan R. Bishop, Holger Fehske, Georg Hager, Olaf Schenk, Jonas Thies, and Gerhard Wellein. 2020. A Recursive Algebraic Coloring Technique for Hardware-Efficient Symmetric Sparse Matrix-Vector Multiplication. *ACM Trans. Parallel Comput.* 7, 3, Article 19 (June 2020), 37 pages.

Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 16–es.

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

Matthias Bollhöfer, Aryan Eftekhari, Simon Scheidegger, and Olaf Schenk. 2019. Large-scale Sparse Inverse Covariance Matrix Estimation. *SIAM Journal on Scientific Computing* 41, 1 (2019), A380–A401.

Matthias Bollhöfer, Olaf Schenk, Radim Janalik, Steve Hamm, and Kiran Gullapalli. 2020. State-of-the-Art Sparse Direct Solvers. (2020), 3–33.

Tyson Brochu and Robert Bridson. 2009. Robust Topological Operations for Dynamic Explicit Surfaces. *SIAM Journal on Scientific Computing* 31, 4 (2009), 2472–2493.

Chris J Budd, Weizhang Huang, and Robert D Russell. 2009. Adaptivity with moving grids. *Acta Numerica* 18 (2009), 111–241.

Nuttapong Chentanez, Ron Alterovitz, Daniel Ritchie, Lita Cho, Kris K. Hauser, Ken Goldberg, Jonathan R. Shewchuk, and James F. O'Brien. 2009. Interactive Simulation of Surgical Needle Insertion and Steering. *ACM Trans. Graph.* 28, 3, Article 88 (July 2009), 10 pages.

Fang Da, Christopher Batty, and Eitan Grinspun. 2014. Multimaterial Mesh-Based Surface Tracking. *ACM Trans. on Graphics (SIGGRAPH North America 2014)* (2014).

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. 2001. Dynamic Real-Time Deformations Using Space and Time Adaptive Sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 31–36.

Leszek Demkowicz. 2006. *Computing with hp-ADAPTIVE FINITE ELEMENTS.* Chapman and Hall/CRC.

Marion Dunyach, David Vanderhaeghe, Loïc Barthe, and Mario Botsch. 2013. Adaptive Remeshing for Real-Time Mesh Deformation. In *Eurographics 2013 - Short Papers*, M.-A. Otaduy and O. Sorkine (Eds.). The Eurographics Association.

Tobias Erhart, Wolfgang A. Wall, and Ekkehard Ramm. 2006. Robust adaptive remeshing strategy for large deformation, transient impact simulations. *Internat. J. Numer. Methods Engrg.* 65, 13 (2006), 2139–2166.

Zachary Ferguson et al. 2020. *IPC Toolkit.* https://ipc-sim.github.io/ipc-toolkit/

Eitan Grinspun, Petr Krysl, and Peter Schröder. 2002. CHARMS: A Simple Framework for Adaptive Simulation. *ACM Trans. Graph.* 21, 3 (July 2002), 281–290.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

David Hahn and Chris Wojtan. 2015. High-Resolution Brittle Fracture Simulation with Boundary Elements. *ACM Trans. Graph.* 34, 4, Article 151 (July 2015), 12 pages.

Kai Hormann, Günther Greiner, and Swen Campagna. 1998. Hierarchical Parametrization of Triangulated Surfaces. *Proceedings of Vision, Modeling and Visualization* (Jan. 1998).

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages.

Dave Hutchinson, Martin Preston, and Terry Hewitt. 1996. Adaptive Refinement for Mass/Spring Simulations. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '96* (Poitiers, France). Springer-Verlag, Berlin, Heidelberg, 31–45.

Zhongshi Jiang, Jiacheng Dai, Yixin Hu, Yunfan Zhou, Jeremie Dumas, Qingnan Zhou, Gurkirat Singh Bajwa, Denis Zorin, Daniele Panozzo, and Teseo Schneider. 2022. Declarative Specification for Unstructured Mesh Editing Algorithms. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 41, 6, Article 251 (Nov. 2022), 14 pages.

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph.* 36, 6, Article 186 (Nov. 2017), 9 pages.

Bryan M. Klingner, Bryan E. Feldman, Nuttapong Chentanez, and James F. O'Brien. 2006. Fluid Animation with Dynamic Meshes. In *ACM SIGGRAPH 2006 Papers* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 820–825.

Woojong Koh, Rahul Narain, and James F. O'Brien. 2015. View-Dependent Adaptive Cloth Simulation with Buckling Compensation. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (Oct. 2015), 1138–1145.

Dan Koschier, Sebastian Lipponer, and Jan Bender. 2015. Adaptive Tetrahedral Meshes for Brittle Fracture Simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 57–66.

Rolf Krause and Patrick Zulian. 2016. A Parallel Approach to the Variational Transfer of Discrete Fields between Arbitrarily Distributed Unstructured Finite Element Meshes. *SIAM Journal on Scientific Computing* 38, 3 (2016), C307–C333.

Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E. Brown, and Laurence Boissieux. 2018. An Implicit Frictional Contact Solver for Adaptive Cloth Simulation. *ACM Trans. Graph.* 37, 4, Article 52 (July 2018), 15 pages.

Ling Li and Vasily Volkov. 2005. Cloth Animation with Adaptively Refined Meshes. In *Proceedings of the Twenty-Eighth Australasian Conference on Computer Science - Volume 38* (Newcastle, Australia) *(ACSC '05)*. Australian Computer Society, Inc., AUS, 107–113.

Minchen Li, Zachary Ferguson, Teseo Schneider, Chenfanfu Jiang, Denis Zorin, Daniele Panozzo, and Danny M. Kaufman. 2023. Convergent Incremental Potential Contact. arXiv.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection-and Inversion-Free, Large-Deformation Dynamics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4, Article 49 (Aug. 2020), 20 pages.

Richard Lipton, Donald Rose, and Robert Targan. 1979. Generalized Nested Dissection. *SIAM J. Numer. Anal.* 16, 2 (1979), 346–358.

S. Léger, A. Fortin, C. Tibirna, and M. Fortin. 2014. An updated Lagrangian method with error estimation and adaptive remeshing for very large deformation elasticity problems. *Internat. J. Numer. Methods Engrg.* 100, 13 (2014), 1006–1030.

Pierre-Luc Manteaux, Wei-Lun Sun, Francois Faure, Marie-Paule Cani, and James F. O'Brien. 2015. Interactive Detailed Cutting of Thin Sheets. In *Proceedings of ACM SIGGRAPH Motion in Games.* 1–8.

P.-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M.-P. Cani. 2017. Adaptive Physically Based Models in Computer Graphics. *Computer Graphics Forum* 36, 6 (2017), 312–337.

Sandeep Menon, Kyle G. Mooney, K.G. Stapf, and David P. Schmidt. 2015. Parallel adaptive simplical re-meshing for deforming domain CFD computations. *J. Comput. Phys.* 298 (2015), 62–78.

Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. 2014. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE Transactions on Visualization and Computer Graphics* 20, 1 (2014), 4–16.

Marek Krzysztof Misztal and Jakob Andreas Bærentzen. 2012. Topology-Adaptive Interface Tracking Using the Deformable Simplicial Complex. *ACM Trans. Graph.* 31, 3, Article 24 (jun 2012), 12 pages.

William F. Mitchell. 1991. Adaptive refinement for arbitrary finite-element spaces with hierarchical bases. *J. Comput. Appl. Math.* 36, 1 (1991), 65–78. Special Issue on Adaptive Methods.

William F. Mitchell and Marjorie A. McClain. 2014. A Comparison of Hp-Adaptive Strategies for Elliptic Partial Differential Equations. *ACM Trans. Math. Softw.* 41, 1, Article 2 (Oct. 2014), 39 pages.

J. F. Molinari and M. Ortiz. 2002. Three-dimensional adaptive meshing by subdivision and edge-collapse in finite-deformation dynamic–plasticity problems with application to adiabatic shear banding. *Internat. J. Numer. Methods Engrg.* 53, 5 (2002), 1101–1126.

J. Mosler and M. Ortiz. 2007. Variational h-adaption in finite deformation elasticity and plasticity. *Internat. J. Numer. Methods Engrg.* 72, 5 (2007), 505–523.

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

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

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

James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. 2002. Graphical Modeling and Animation of Ductile Fracture. *ACM Trans. Graph.* 21, 3 (July 2002), 291–294.

James F. O'Brien and Jessica K. Hodgins. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of ACM SIGGRAPH 1999.* ACM Press/Addison-Wesley Publishing Co., 137–146.

Maria Elizabeth G. Ong. 1994. Uniform Refinement of a Tetrahedron. *SIAM Journal on Scientific Computing* 15, 5 (1994), 1134–1144.

Jean-Paul Pelteret. 2016. The 'Quasi-Static Finite-Strain Compressible Elasticity' code gallery program. https://dealii.org/developer/doxygen/deal.II/code_gallery_Quasi_static_Finite_strain_Compressible_Elasticity.html. Accessed: 2023-04-24.

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, Article 110 (July 2014), 9 pages.

Devon Powell. 2021. PolyClipper. https://github.com/LLNL/PolyClipper.

Alfred Schmidt and Kunibert G. Siebert. 2000. A posteriori estimators for the h − p version of the finite element method in 1D. *Applied Numerical Mathematics* 35, 1 (2000), 43–66.

Teseo Schneider, Jérémie Dumas, Xifeng Gao, Denis Zorin, and Daniele Panozzo. 2019. *PolyFEM.* https://polyfem.github.io/

Timothy J. R. Simnett, Stephen D. Laycock, and Andy M. Day. 2009. An Edge-based Approach to Adaptively Refining a Mesh for Cloth Deformation. In *Theory and Practice of Computer Graphics*, Wen Tang and John Collomosse (Eds.). The Eurographics Association.

Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. 2014. Designing Inflatable Structures. *ACM Trans. Graph.* 33, 4, Article 63 (July 2014), 10 pages.

Jonas Spillmann and Matthias Teschner. 2008. An Adaptive Contact Model for the Robust Simulation of Knots. *Computer Graphics Forum* 27, 2 (2008), 497–506.

Keith Stein, Tayfun E. Tezduyar, and Richard Benney. 2004. Automatic mesh update with the solid-extension mesh moving technique. *Computer Methods in Applied Mechanics and Engineering* 193, 21 (2004), 2019–2032. Flow Simulation and Modeling.

Vasileios Vavourakis, Dimitrios Loukidis, Dimos C. Charmpis, and Panos Papanastasiou. 2013. Assessment of Remeshing and Remapping Strategies for Large Deformation Elastoplastic Finite Element Analysis. *Comput. Struct.* 114–115 (Jan. 2013), 133–146.

J. Villard and H. Borouchaki. 2005. Adaptive Meshing for Cloth Animation. *Eng. with Comput.* 20, 4 (Aug. 2005), 333–341.

Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. 2010. Dynamic Local Remeshing for Elastoplastic Simulation. *ACM Trans. Graph.* 29, 4, Article 49 (July 2010), 11 pages.

Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. 2009. Deforming Meshes That Split and Merge. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 76, 10 pages.

Jiayi Eris Zhang, Jèrèmie Dumas, Yun (Raymond) Fei, Alec Jacobson, Doug L. James, and Danny M. Kaufman. 2022. Progressive Simulation for Cloth Quasistatics. *ACM Trans. Graph.* 41, 6, Article 218 (2022).

M. G. Zielonka, M. Ortiz, and J. E. Marsden. 2008. Variational r-adaption in elastodynamics. *Internat. J. Numer. Methods Engrg.* 74, 7 (2008), 1162–1197.