Precomputed Mesh Optimization Using Modal Analysis for Elastic Body Simulations

Nobuo Nakagawa
The University of Tokyo
Tokyo, Japan
nobuo-nakagawa@g.ecc.u-tokyo.ac.jp

Takashi Kanai
The University of Tokyo
Tokyo, Japan
kanait@acm.org

Abstract

We propose a pre-computation pipeline for generating low-resolution meshes optimized for elastic body behavior, aiming to balance computational efficiency and shape fidelity in elastic body simulations. The method begins with modal analysis on the input mesh to estimate plausible deformation shapes based on dominant low-frequency vibration modes. The resulting strain distributions are then computed, and adaptive mesh simplification is achieved by selectively applying edge contraction according to the spatial distribution of strain, allowing for locally controlled resolution. To preserve geometric features and generate meshes suitable for elastic simulation, the mesh decimation process incorporates not only standard Quadric Error Metrics (QEM), but also triangle quality measures tailored for robust simulation and curvature information from the input mesh. All expensive steps are performed offline during preprocessing, whereas the online simulation runs at reduced cost while preserving the intended deformation behavior. Evaluations across multiple models demonstrate that our approach achieves a favorable trade-off between efficiency and accuracy compared to conventional methods.

Keywords: ***Elastic body simulation, Modal analysis, Adaptive mesh simplification, Quadric Error Metrics (QEM)***

1. Introduction

In real-time physics-based animation, high-resolution simulations using detailed models are desirable to enhance visual realism. However, this comes at the cost of significantly increased computational cost. For elastic deformation, a widely used approach is to perform physical simulation on a coarse proxy mesh [35] and animate the visual mesh based on the simulation results. However, inconsistencies between the proxy and visual meshes can lead to the loss of fine contact details. In regions where the proxy mesh lacks sufficient resolution for accurate deformation, important deformation features may also be lost. A further issue arises when high-resolution mesh elements are allocated to regions with minimal deformation, which may degrade the convergence of the simulation.

To ensure visual fidelity, it is essential to generate meshes that are well-suited for elastic deformation. However, this task is significantly more complex and labor-intensive than simplifying static meshes that do not undergo deformation. In this work, we propose an automated, precomputation-driven pipeline for generating deformation-aware meshes. Achieving this goal requires addressing two major challenges. First, there is no clear definition of what constitutes a mesh suitable for elastic deformation. Second, the process of optimizing a mesh for deformation must not compromise its visual fidelity. To the best of our knowledge, no existing method simultaneously addresses these two conflicting objectives.

The framework proposed in this paper takes as input an arbitrary triangular surface mesh, as commonly used in real-time applications, and automatically optimizes it into a form suitable for elastic deformation, ultimately producing an optimized triangular surface mesh as output. The proposed method consists of two stages: a pre-computation stage that generates the reference data required for mesh optimization, and an optimization stage that uses this data to coarsen the mesh.

In the reference data generation phase, we perform modal analysis to extract representative low-frequency modes. Based on these mode shapes, we estimate typical elastic deformations and compute the corresponding strain distributions. In addition, the curvature data extracted from the input mesh is stored with the strain data as reference information for the next phase. In the optimization phase, we utilize the previously computed strain and curvature data to perform mesh simplification based on Quadric Error Metrics (QEM) [9]. Regions with high strain are considered to require fine deformation detail, and thus mesh decimation is suppressed in these areas. Similarly, regions with high curvature are regarded as geometrically significant and

are also preserved during simplification, enabling adaptive mesh optimization. Furthermore, we evaluate the triangle quality before and after each vertex pair contraction and incorporate the results into the edge selection priority, enabling high-quality remeshing.

Benchmarks on multiple meshes indicate competitive accuracy at reduced cost relative to conventional methods. We also discuss limitations and prospective extensions.

In summary, the contributions of this work are as follows:

- We propose a pipeline that automatically simplifies meshes for elastic deformation by estimating natural deformation modes via modal analysis and computing strain distributions based on those modes.
- We extend a QEM-based mesh simplification method by integrating strain, triangle quality, and curvature information, enabling topology-preserving simplification that is suitable for simulation while retaining geometric features of the input mesh.
- We validate the effectiveness and computational efficiency of the proposed method through experiments on multiple models.

2. Related Works

In this section, we will discuss existing research closely related to this research, divided into subsections by topic.

2.1. Deformable Objects

Since the pioneering work of Terzopoulos et al. [32], considerable effort has been directed toward simulating the motion of deformable objects. An early comprehensive survey of this topic is provided in [29]. Provot [27] proposed constraints to control the stiffness of the mass-spring model while maintaining its flexibility. This paper led to the widespread adoption of the mass-spring model as an early technique for cloth simulation. Baraff and Witkin [1] improved the mass-spring model and introduced an implicit integration technique to stably simulate stiff cloth, making model more efficient and stable. Müller et al. [24] proposed Position-Based Dynamics (PBD) as an alternative to the mass-spring model, which allows for more intuitive and stable simulation by handling stiffness and constraints at the position level. Macklin et al. [18] proposes a parallel constraint solver based on PBD, enabling unified simulation of fluids, flexible bodies, rigid bodies, cloth, etc. in real time. XPBD [17] inherits the stability and ease of use of PBD, while resolving the issue of stiffness values depending on the number of iterations, and makes it possible to control natural stiffness values by specifying real-world compliance values. SubstepXPBD [19] showed that constraint error and numerical damping can be significantly reduced by dividing the time step into smaller steps and using one constraint solver iteration for each smaller step, rather than using multiple constraint solver iterations for a single large time step. In this study, we adopt SubstepXPBD to simulate elastic deformation of the mesh for stability.

2.2. Modal Analysis

Modal analysis in physics-based animation was pioneered by Pentland and Williams [26], who demonstrated its effectiveness in computing linear elastic body deformations, thereby enabling enhanced and exaggerated animations. This foundational work was subsequently extended by Barbič and James [2], who developed a framework supporting non-linear deformations with real-time computational capabilities. While subspace approaches such as Barbič and James [2] reduce the *state dimension* by projecting dynamics onto a low-dimensional basis, our method is orthogonal to this direction: we reduce the *spatial degrees of freedom* through topology-preserving mesh simplification performed as an offline preprocessing step. Both strategies aim to accelerate elastic simulations but operate on different layers of the pipeline, and they could be composed in practice. Particularly relevant to our research is the work of Sellán et al. [30], which employs modal analysis to generate precomputed fracture patterns. However, their investigation was primarily confined to fracture geometry pre-generation.

2.3. Remeshing

Khan et al. [15] and Luebke [16] provides a comprehensive survey of remeshing up to recent years. Although there are many topics related to remeshing, we will explain mesh simplification, feature preservation, and adaptive remeshing, which are closely related to our research.

2.3.1 Mesh Simplification

As a fundamental approach to mesh simplification, Hoppe [12] proposed a new mesh data structure called Progressive Meshes, which uses edge-collapse operations to simplify meshes while recording their hierarchical structure, thereby allowing them to be reconstructed at any desired level of detail (LOD). In subsequent work, Hoppe [13] improved Progressive Meshes through selective refinement and proposed a method for dynamically adjusting local mesh detail based on the viewpoint and the resolution of the screen-projected surface. Meanwhile, Garland and Heckbert [9] introduced the Quadric Error Metric (QEM) and developed a method to mathematically minimize shape error during vertex contraction operations. Garland later expanded on this QEM-based polygonal surface simplification technique in greater detail in his doctoral dissertation [8].

Table 1: Comparison to Existing Methods

Methods	Preserve topology	Precomputed	Modal analysis	Focus
[30]	-	+	+	Fracture patterns
[35]	-	+	-	Proxy mesh for cloth
[7], [33]	+	-	-	In-Timestep remeshing
[21]	-	+	-	Animation-driven quadrangulation
Ours	+	+	+	Elastic-body optimization

One of the important concepts introduced in this research is *area-weighted* QEM, where the metric is normalized based on the area of each triangle. This area-based normalization is intended to provide more robust and consistent simplification quality, even for models with non-uniform input mesh tessellation density or triangle sizes. In our current research, we adopt this area-weighted QEM as a fundamental error metric and apply further original extensions to achieve effective mesh simplification. Hu et al. [14] proposed an error-bounded, feature-preserving surface remeshing method that treats the symmetric Hausdorff distance δ as a hard constraint while greedily improving the worst interior angle and seeking the coarsest mesh above a target θ; quality is driven by a priority-queue scheduling of local operators (collapse/relocate/split), with feature-preserving relocations. In contrast, we steer simplification by prospective deformation (modal strain) and curvature within a QEM-based objective to favor elastodynamic fidelity rather than enforcing a strict Hausdorff bound. Micro-Mesh Construction [20] converts multi-million-triangle inputs into a compact μ-mesh representation designed for native, hardware-accelerated ray tracing, where each coarse base triangle is subdivided and displaced at render time. Their method tailors simplification to tessellation/displacement sampling, enforcing isotropy and reprojectability and explicitly reducing prismoid volume; in contrast, we optimize meshes for elastic simulation quality rather than rendering acceleration.

2.3.2 Feature Preservation

In mesh simplification and remeshing, preserving important geometric features such as sharp edges and high-curvature regions is crucial for maintaining the visual quality of models. Dassi et al. [4] proposed a curvature-adaptive remeshing method that embeds surfaces in a high-dimensional space (R^6) and estimates mesh size and quality based on curvature, thereby generating high-quality meshes while preserving sharp features using local operations. Recent research continues to emphasize the use of curvature information for feature preservation. For instance, Su et al. [31] extended conformal mapping techniques and proposed curvature-adaptive remeshing by sampling normal cycles instead of the surface itself, aiming for faithful geometry preservation and efficient feature extraction. While acknowledging these sophisticated approaches, our research prioritizes ease of implementation and computational efficiency. In our QEM-based mesh simplification framework, we aim to incorporate simplified features derived from vertex normal information to influence QEM prioritization. This approach seeks to maintain a degree of geometric feature preservation while suppressing computational costs. Compared to the advanced methods discussed above, our approach is more straightforward and aims to achieve a practical balance between computational speed and output quality.

2.3.3 Adaptive Remeshing

We distinguish two complementary axes for adaptivity: (i) runtime remeshing that optimizes connectivity during simulation, and (ii) field-driven geometric guidance that steers offline meshing objectives.

Adaptive remeshing is a critical technique for balancing accuracy and computational efficiency in simulations involving complex physical phenomena. Addressing a broad range of material behaviors, Wicke et al. [34] proposed dynamic local remeshing in material space for elastoplastic simulations, developing a method to maintain mesh quality while suppressing artificial plasticity and numerical diffusion. Ferguson et al. [7] introduce in-timestep remeshing for contacting elastodynamics, using energy change as a per-timestep criterion; Wen et al. [33] then formulate an optimal r-adaptive variant that jointly optimizes remeshing and the implicit state, matching or exceeding much finer fixed meshes with far fewer DOF. For task-specific adaptivity, Narain et al. [25] drive anisotropic remeshing by curvature and velocity to capture cloth wrinkles. Furthermore, for interactive applications, Dunyach et al. [5] proposed a method that utilizes a curvature-adaptive sizing field to achieve both real-time performance and high-quality mesh deformation. These advanced methods share the common goal of improving accuracy, stability, or interactivity by dynamically optimizing the mesh during simulation runtime.

Orthogonal to runtime adaptivity, there is a line of field-driven geometric guidance. Animation-Aware Quadrangulation [21] , for example, aggregates per-frame stretch over

an input animation to derive a deformation-driven guidance field and aligns a cross field with principal directions for quadrangulation.

In contrast, our method precomputes a prospective guidance field from dominant modal strain on a single static mesh and uses it to bias QEM-based edge collapses while preserving topology. This avoids runtime remeshing costs yet concentrates resolution in deformation-salient regions, as detailed in Section 3.

2.4. Comparison to Existing Methods

Table 1 contrasts recent approaches along three axes—topology preservation, precomputation, and use of modal analysis. Representative entries are: Sellán et al. [30], which address precomputed modal fracture; Zheng et al. [35], which generate cloth proxy meshes with altered topology; Ferguson and Wen [7, 33], which adapt meshes online each timestep (r-adaptive ITR); and Animation-Aware Quadrangulation [21], which derives a deformation-driven guidance field from animated sequences for field-aligned quadrangulation. In contrast, our method is an offline, modal-guided pipeline that uses *prospective* deformation (low-order modal strain) to steer QEM-based simplification while preserving topology.

In summary, our research is distinct in its focus on precomputed mesh optimization specifically for elastic deformation, with an emphasis on maintaining topology.

3. Method

We adopt a two-phase *offline* pipeline to produce an adaptively resolved mesh optimized for elastodynamic simulation (see Figures 1, 3):

- **Phase 1: Modal Analysis** (Algorithm. 1). On the input triangular mesh, we compute low-frequency modes and form a scalar, per-triangle prospective strain signal; in parallel we precompute mean curvature.
- **Phase 2: Mesh Optimization** (Algorithm. 2). Guided by the prospective strain and curvature, we perform QEM-based edge collapses with a triangle-quality prior to obtain the simplified mesh.

Note that our pipeline outputs a single, topology-preserving mesh with *spatially varying resolution*; we do not perform runtime adaptive remeshing nor alter connectivity during simulation.

All computations are performed prior to simulation. The details of the modal analysis and the optimization objective are given in Sections 3.1 and 3.2, respectively.

3.1. Modal analysis

Modal analysis is a method to express a complex vibration system as a superposition of simple vibration modes. In this paper, we utilize modal analysis to pre-determine representative deformation shapes that an elastic body is expected to exhibit, and these shapes are then used as reference information for mesh optimization. The equation of motion for a multi-degree-of-freedom system, omitting the damping term, is expressed as follows:

$$M\ddot{x}(t) + Kx(t) = f_{ext}(t), \tag{1}$$

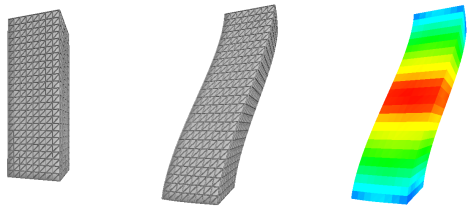
where M is the mass matrix, K is the stiffness matrix, $x(t)$ is the displacement vector, and $f_{ext}(t)$ is the external force vector.

Consider the free vibration of this system ($f_{ext}(t) = 0$) and substitute the assumed solution expressed as a complex exponential function as shown below:

$$x(t) = \phi e^{i\omega t}, \tag{2}$$

where ϕ is the vibration mode (eigenvector), and ω is the natural frequency. Substituting this assumption into the equation of motion and simplifying, we obtain the eigenvalue Equation 3. The condition for Equation 3 to have a non-trivial solution is given by the characteristic Equation 4:

$$(K - \omega^2 M)\phi = 0, \tag{3}$$



(a) input (b) deformation (c) strain

Figure 1: Modal analysis stage. From the input triangular surface mesh (a), we synthesize a low-frequency mode shape (b) and compute per-triangle Green–Lagrange strain; the rank-normalized strain is visualized (c) with high strain in red and low in blue.

Algorithm 1 Overview of modal analysis.

procedure ModalAnalysis($file$) ▷ Phase1
  $mesh \leftarrow$ LoadMesh($file$)
  $curv \leftarrow$ ComputeCurvature($mesh$)
  $M \leftarrow$ BuildMassMatrix($mesh$) ▷ lumped mass
  $K \leftarrow$ BuildStiffnessMatrix($mesh$)
  $E_{val}, E_{vec} \leftarrow$ ComputeEigenValue(M, K)
  $modes \leftarrow$ SelectRepresentativeModes(E_{val}, E_{vec})
  $mode \leftarrow$ UserSelectedModes($modes$)
  $mesh' \leftarrow$ ComputeShape($mesh, mode$) ▷ sin() * mode
  $strain \leftarrow$ ComputeStrain($mesh'$) ▷ Green-Lagrange strain
  return $curv, strain$
end procedure

$$\det(K - \omega^2 M) = 0. \tag{4}$$

By solving this equation, the natural frequencies ω_i and corresponding natural modes ϕ_i of the structure are obtained. An eigenmode ϕ_i represents the deformation shape of a structure when it vibrates at a specific eigenfrequency ω_i. In an N-degree-of-freedom system, there are N eigenfrequencies and their corresponding eigenmodes.

The natural modes and natural frequencies are obtained by constructing and solving the mass matrix M and stiffness matrix K using the input vertices and edges of the triangular surface mesh given as input. The low-frequency mode is considered to be a representative vibration, and the obtained eigenmode value is used to approximately handle the vibration with a simple sine curve approximation to obtain an approximate shape of a representative elastic deformation.

To visualize each low-frequency mode and to define a single prospective signal, we synthesize a displaced configuration from a normalized mode shape $\boldsymbol{\phi}$ as

$$\mathbf{x}'(\theta) = \mathbf{x}_0 + \sin(\theta)\,\boldsymbol{\phi}, \tag{5}$$

where θ controls the phase/amplitude.

In Algorithm 1 we: (i) load the mesh and precompute mean curvature (steps 2–3), (ii) build the lumped mass and stiffness matrices and solve the generalized eigenproblem (steps 4–6), (iii) select a representative low-frequency mode (steps 7–8), (iv) synthesize a displaced configuration (Eq. (5); step 9), and (v) compute the Green–Lagrange strain and store its per-triangle rank as our prospective signal (steps 10–11). Figure 1 illustrates (iv)–(v): the mode shape (b) and the rank-normalized strain map (c).

While our default experiments use a single dominant mode, the same pipeline can optionally use multiple low-frequency modes to build a composite guidance field. Let $E_k(\cdot)$ be the Green–Lagrange strain induced by the k-th mode via Eq. (5). For $r \geq 1$ selected modes with nonnegative weights w_k and aggregation parameter $p \in [1, \infty]$, we define

$$\sigma_r(T) = \Big(\sum_{k=1}^{r} w_k \, \|E_k(T)\|_F^{\,p} \Big)^{1/p}. \tag{6}$$

Here p controls how modes are combined: $p{=}1$ yields a weighted sum, $p{=}2$ an RMS average, and $p{\to}\infty$ a max-envelope; using σ_r simply replaces σ in E_{strain}. However, aggregating many modes is not universally beneficial—when modes emphasize different regions, the guidance can become diluted—so, for clarity and reproducibility, we report results with the single-mode variant ($r{=}1$) and leave robust choices of (r, p, w_k) to future work.

Examples of vibration modes can be visualized as shown in Figure 2. Users can select an appropriate mode depending on the shape and application of the analysis target. Based on the selected mode, the deformed shape is estimated, deformation gradient information from the initial shape is calculated, the Green-Lagrange strain tensor is obtained from the deformation information, and optimal mesh simplification is performed based on the magnitude of the Frobenius norm.

We consider a rectangular bar and visualize its *bending eigenmode*, which produces an S-shaped deflection of the centerline, as shown in Figure 1. Using this deformed shape, we calculated the deformation gradient $F = X'X^{-1}$ for each triangle as the deformed shape X' for the initial shape X, and further scalarized the value using the Frobenius norm of the Green-Lagrange strain tensor $E = \frac{1}{2}(F^T F - I)$ to obtain the strain value for each triangle in Figure 1(c). The strain values are sorted for comparison within the mesh, and then the ranked values among all values within the mesh are stored as values [0,1]. This value becomes reference information for the later mesh optimization phase. The reason for using the rank within the same mesh instead of directly using the strain is that even if triangles have similar strain values, only meshes in appropriate areas are selectively deleted.

The modal analysis used in our method is a linear modal analysis that does not consider damping, which is widely used as a method to obtain natural vibration modes. On the other hand, SubstepXPBD [19] is adopted for the elastic deformation simulation, which is a position-based method that can stably handle elastic deformation. Although the calculation models of the two are different, they have in common that low-frequency deformation components are important in the main deformation patterns of the system. By leveraging these low-order modes in our framework, we can pre-analyze the types of deformation likely to occur in XPBD simulations and perform targeted mesh optimization accordingly.

In addition to the modal analysis, the mean curvature is calculated from the input mesh and stored for each triangle. Since a triangular mesh is used as input, we adopt the method proposed by [22] for estimating mean curvature on discrete surfaces. The curvature is discretized based on the area-weighted average over the one-ring neighborhood of each triangle and stored per triangle for use in later phases. Figure 3 (a) shows the mean curvature calculated for each triangle using a red and blue color map.

3.2. Mesh optimization

In the second phase, the mesh optimization phase, we apply a mesh reduction method based on QEM [9, 8], while referring to the deformation gradient information and mean curvature information obtained in the first phase. We adopt the standard area-weighted QEM to score a candidate edge collapse $e = (v_i, v_j) \to v^\star$:

$$E_{\text{QEM}}(e) = \tilde{v}^{\star\top} \big(Q_{v_i} + Q_{v_j} \big)\, \tilde{v}^\star, \tag{7}$$

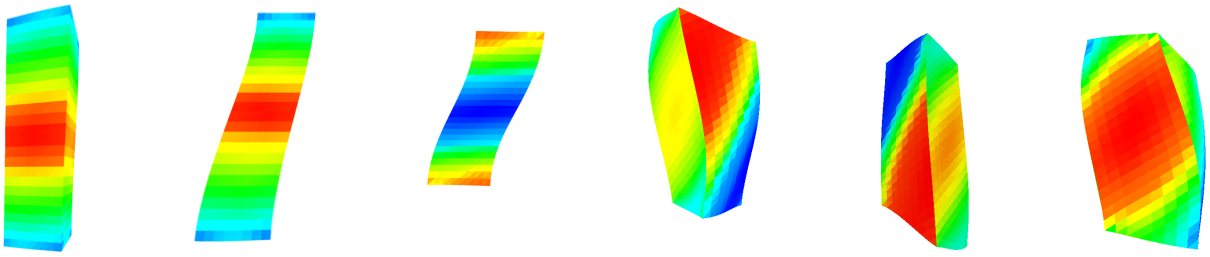
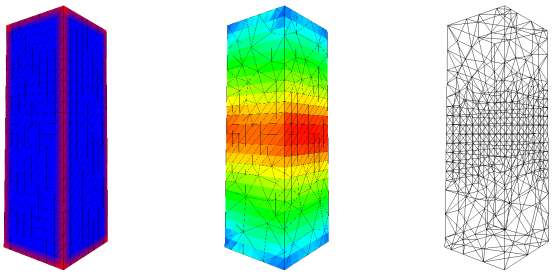


Figure 2: Examples of vibration modes used in this method. Six representative deformed shapes were estimated from different vibration modes. Users can select an appropriate mode depending on the application to obtain the deformed shape. The strain values calculated from the estimated deformed shapes are ranked within the mesh and shown on a map from red to blue.



(a) curvature (b) mesh reduction (c) optimized

Figure 3: Mesh optimization stage. (a) Mean curvature on the input mesh. (b) QEM-based edge collapses biased by prospective strain and curvature. (c) Final simplified, topology-preserving mesh.

Algorithm 2 Overview of mesh optimization

procedure MESHOPTIMIZE($mesh, strain, curv$) ▷ Phase2
  for each vertex $v \in$vertices **do**
    $v.quadric \leftarrow 0$ ▷ initialize
  end for
  for each triangle $tri \in$triangles **do**
    $Kp \leftarrow$ BUILDQUADRICMATRIX(tri)
    for each vertex $v \in$tri **do**
      $v.quadric \leftarrow Kp$
    end for
  end for
  for each edge $e \in$edges **do**
    $score \leftarrow Score(edge, strain, curv)$ ▷ + triangle_q
    $heap.push(e, score)$
  end for
  $best = heap.top()$ ▷ best candidate
  $EdgeCollapse(best)$
end procedure

where Q_v is the area-weighted sum of plane quadrics of faces incident to v and $\tilde{v}^\star = (x^\star, y^\star, z^\star, 1)^\top$ is the homogeneous coordinate of the collapsed vertex.

The mesh optimization method proposed in our approach extends vertex pair contraction based on QEM to generate meshes suitable for elastic deformation. Specifically, the QEM for each vertex is first calculated, and the pair contraction cost is evaluated. This cost function, C, is defined as a linear combination of the standard QEM error term, the strain pre-calculated by modal analysis, and the mesh curvature ($curv$), as expressed in the following equation:

$$C = E_{QEM} + \alpha \cdot E_{strain} + \beta \cdot E_{curv}, \tag{8}$$

where E_{QEM} represents the error from QEM, E_{strain} is the cost based on strain, and E_{curv} is the cost based on curvature. α and β are user-specifiable weighting factors to adjust the contribution of each term.

For a candidate edge collapse $e = (v_i, v_j) \to v^\star$, we evaluate the terms in Eq. (8) from fields precomputed in Phase 1. Let $\sigma : \mathcal{T} \to \mathbb{R}_{\geq 0}$ denote the per-triangle prospective-strain magnitude and $\kappa : \mathcal{V} \to \mathbb{R}_{\geq 0}$ the per-vertex mean-curvature magnitude on the input mesh. Using kd-tree queries to the nearest input triangle $\pi_T(\cdot)$ and vertex $\pi_V(\cdot)$, we set

$$E_{\text{strain}}(e) = \max\{\sigma(\pi_T(v_i)),\, \sigma(\pi_T(v_j))\},$$
$$E_{\text{curv}}(e) = \kappa(\pi_V(v^\star)).$$

These lookups are $O(\log n)$ and avoid recomputing strain/curvature during decimation; the weights α, β in Eq. (8) control their relative influence. Scoring a candidate collapse and pushing it to the heap follow *Algorithm. 2, steps 11–14*, and the iterative contraction/update loop is *Alg. 2, steps 15–16*.

Regions with high strain values are expected to undergo significant elastic deformation, thus requiring the preservation of detailed mesh information. Conversely, areas with

low strain values are considered to undergo relatively less detailed deformation, allowing for mesh reduction to decrease the overall face count while maintaining visual fidelity. Additionally, incorporating the input mesh's curvature into the cost function is intended to preserve the sharp features of the original geometry. Our method employs the normalized QEM proposed in [8], enabling consistent QEM calculations independent of mesh tessellation. Based on the cost function C in Equation 8, the mesh is adaptively simplified to the target polygon count by iteratively contracting the vertex pair with the minimum cost and updating the costs of affected neighboring pairs. This adaptive simplification process continues until the target polygon count is achieved. Areas with high strain indicate the application of large forces and significant elastic deformation, necessitating detailed mesh representation. In contrast, areas with low strain can be interpreted as undergoing relatively minor deformation, allowing simplification without compromising visual plausibility. In contrast to prior distance field-based proxy mesh generation approaches [35], our method performs QEM-based mesh simplification, inherently maintaining the original mesh topology.

3.2.1 Triangle quality

In mesh optimization for elastic body simulation, it is desirable to maintain high-quality meshes by avoiding the generation of sliver triangles, which can adversely affect simulation accuracy. Therefore, an approach was adopted to evaluate triangle mesh quality before and after vertex pair contraction operations. As quality evaluation methods, the technique using the cosine of the internal angles of a triangle, proposed in [11], and the formulation of triangle quality proposed in [23] were compared. After examining the score histograms for 10,000 randomly generated triangles, the method from [23] was adopted due to its favorable value distribution.

$$q_{triangle} = \frac{4\sqrt{3}\,A}{\ell_1^2 + \ell_2^2 + \ell_3^2}, \qquad \text{as in [23]}, \tag{9}$$

$$E_{\text{tri}_\text{q}} = 1 - q_{triangle}. \tag{10}$$

Here, A is the area of triangle, and ℓ_1, ℓ_2, ℓ_3 are the edge lengths of triangle.

By incorporating this quality measure into the cost function defined in the previous section (Equation 8), mesh reduction that also considers triangle quality is performed. The final cost function C is defined as follows:

$$C = E_{QEM} + \alpha \cdot E_{strain} + \beta \cdot E_{curv} + \gamma \cdot E_{tri\_q}. \tag{11}$$

This term is folded into the per-edge score before heap insertion in *Algorithm. 2, step 12*. The procedure is as follows: (i) initialize per-vertex quadrics, (ii) accumulate plane quadrics over incident faces, (iii) score every edge with the composite cost (Eq. 11) that combines QEM, prospective strain, curvature, and triangle-quality, (iv) push candidates to a heap and repeatedly pop the best, collapse it, and update affected neighbors until the target budget is met. See Algorithm 2.

Weights. For all experiments, we use fixed coefficients of $\alpha = 0.05$, $\beta = 1.0$, and $\gamma = 0.01$ in Eq. (11). These values were empirically chosen to balance the relative contributions of the strain, curvature, and triangle-quality terms. The fields E_{strain} and E_{curv} are normalized to $[0, 1]$, so the relative magnitudes of α and β mainly reflect their intended importance (favoring curvature preservation), while γ provides mild regularization to maintain mesh element quality. All reported results use the same parameter setting without further tuning.

4. Implementation

We implemented our algorithm in C++, using Eigen [10] and Spectra [28] for linear–algebra routines. Spectra is employed to accelerate the modal analysis of large meshes by efficiently solving sparse (generalized) eigenvalue problems; when such acceleration is unnecessary, an Eigen-only implementation suffices.

All experiments integrate SubstepXPBD [19] on the *triangular surface mesh*. We use standard distance (edge-stretch) constraints with substepping; no runtime remeshing or connectivity changes are performed. Thus, our cloth-/shell-like behavior is modeled via XPBD-style constraints rather than a thin-shell FEM formulation. A volumetric (tetrahedral) XPBD variant was developed but *not* used in the reported results.

For generality, modal analysis is computed in a free–free setting (no gravity and no fixed supports, i.e., no Dirichlet/clamped DOFs) to obtain boundary-agnostic low-frequency modes that reflect the intrinsic elastic response of the shape. At runtime, we simulate with task-specific conditions (e.g., bottom clamped with a prescribed twist). This mismatch can reduce optimality when deformation is dominated by task-specific constraints or contacts. In practice we preview the first few non-rigid, low-frequency modes and select one whose strain saliency aligns with the intended use (e.g., dominant bending or sway). Although the modal and runtime boundary conditions differ, the prospective strain hotspots were qualitatively stable across BC choices in our examples and sufficed to steer coarsening.

In our experiments, we previewed the first few non-rigid, low-frequency modes and empirically selected one whose strain saliency aligned with the intended deformation type (e.g., bending, twisting). This manual selection was used only to demonstrate the feasibility of mode-guided simplification, rather than to claim it as an optimal strategy.

All experiments were performed on a machine with an

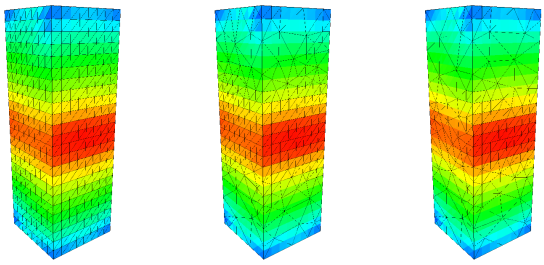


Figure 4: From left to right, the mesh optimization results are shown with an increasing number of edge contractions. The user can interactively change the number of edge contractions to any desired value and check the results. The strain values are shown in the color map, but the curvature values can also be displayed in the color map instead.

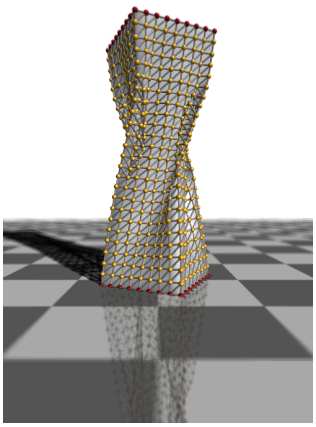


Figure 5: Twisted Rectangle (Before Optimization).

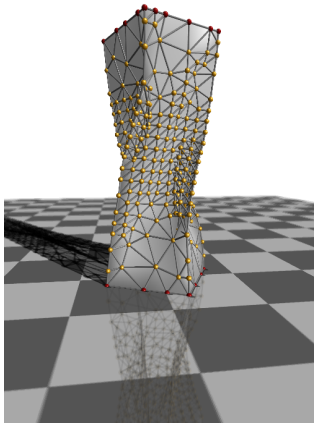


Figure 6: Twisted Rectangle (After Optimization).

Intel® Core™ i7–8700B CPU (3.20 GHz) and 32 GB of RAM. The results reported in this paper (e.g., Figures 1, 3, 4, 5, and 6) were reproduced on Windows, macOS, and Linux from the same codebase. Under these conditions, our implementation attains interactive simulation rates even for elastodynamic simulations on comparatively complex models.

Our reference implementation used to generate all results will be released as an open source project.

5. Results

Table 2 reports, for each model, the face/vertex counts before and after optimization together with the *per-step simulation time on CPU*. Figures 5 and 6 show elastic-body simulations on the optimized mesh, derived from a basic rectangular parallelepiped mesh. The simulations were performed with SubstepXPBD [19] using 2,000 substeps. Owing to the reduced number of degrees of freedom, the optimized (yet topology-preserving) mesh achieves faster per-step runtimes. At the same time, the decimation is spatially varying: regions with high prospective strain and/or curvature retain more sampling, whereas low-activity regions are coarsened. This non-uniform resolution helps preserve visual plausibility despite simplification. The observed speedup stems from a reduced number of spatial DOF in the full-order solver; we do not claim runtime advantages over reduced-order (subspace) integrators [2], which address a different layer of acceleration.

We report *per-step CPU runtime* of the elastodynamic solver (SubstepXPBD, 2,000 substeps). All timings are taken on the same hardware and with identical solver settings (time step, constraints). Precomputation (modal analysis, curvature, kd-trees, QEM setup) is performed offline and excluded from these online metrics; where available, we report precomputation time separately. For each model we also report the relative speedup

$$S = \frac{t_{\text{baseline}}}{t_{\text{optimized}}} \tag{12}$$

where t_{baseline} is the per-step runtime on the original mesh and $t_{\text{optimized}}$ on our simplified mesh.

Unlike the S-shaped deformation based on vibration modes, a twisted shape is compared, but an effective simulation is performed because the spatially varying resolution division of the mesh prioritizes areas where detailed information is required, since the low-frequency vibration modes represent the typical shapes of elastic deformations.

As an example, we present a non-uniform mesh in the shape of the Eiffel Tower, taken from the Thingi10K [36] dataset. The modal analysis and mesh optimization phases are shown in Figures 7 and 8, respectively. Figures 9 and 10 present the results of elastic body simulations using the mesh before and after optimization.

Since mesh optimization is performed in advance, it is possible to perform high-speed simulation of elastic deformation without requiring CPU time at runtime. In addition, since the workflow is automated, users can obtain optimized meshes without editing the mesh. On the other hand, if users are not satisfied with the automatically generated mesh shape, they can further improve the mesh quality by changing the threshold used for the strain or specifying the extent to which curvature information is referenced.

6. Discussion

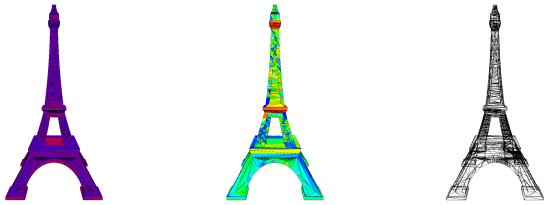
In this section, we first validate the contribution of each component of our cost function through an ablation study. We then discuss the effectiveness and uniqueness of our method through comparisons with existing research. Fi-

Table 2: Model profiles and online simulation performance.

Model	Faces		Vertices		Online Simulation Time (ms, per step, CPU)		
	Original	Optimized	Original	Optimized	Original	Optimized	Speedup S
Rectangle	1,792	950	898	474	13.6	8.0	$\times 1.70$
Eiffel	6,156	3,832	3,072	1,910	48.0	34.0	$\times 1.41$

(a) input (b) deformed shape (c) strain values

Figure 7: This figure presents the modal analysis phase in a format similar to Figure 1. The model shown here is an Eiffel Tower model from the Thingi10K dataset.



(a) curvature (b) mesh reduction (c) optimized

Figure 8: This figure presents the modal mesh optimization phase in a format similar to Figure 3. The model shown here is an Eiffel Tower model from the Thingi10K dataset.

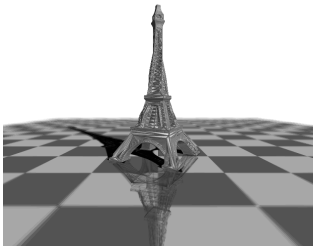


Figure 9: Twisted Eiffel (Before Optimization).

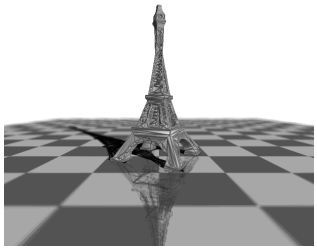


Figure 10: Twisted Eiffel (After Optimization).

nally, we clarify the limitations of this study and discuss possibilities for future extensions.

6.1. Ablation Study

To validate the effectiveness of each term in our cost function, we conducted an ablation study. Figure 11 visually compares optimization results on the *Rectangle* model as we incrementally add terms, with all outputs simplified to the same face budget (950 faces). As a baseline, using *pure* area-weighted QEM (a) produces nearly uniform simplification. To avoid confusion with widely used MeshLab [3] results, we clarify that the default "Quadric Edge Collapse Decimation" in MeshLab includes a quality regularizer; when this regularizer is disabled, the MeshLab output is indistinguishable from our QEM baseline in our tests. Adding the curvature term (b) preserves resolution along geometric features such as sharp edges and corners. Including the triangle-quality term (c) suppresses sliver triangles, directly improving suitability for numerically stable simulation. Finally, adding the strain term (d) allocates resolution to deformation-dense zones (e.g., the center) while maintaining overall mesh quality. These results indicate that all components we introduce are necessary to balance geometric fidelity, deformation relevance, and element quality.

To quantitatively support the visual comparison in Fig. 11, we summarize the results of each method in Table 3. The table reveals that using only QEM, or adding the curvature term, results in a minimum angle of 0.00 degrees, indicating the creation of degenerate triangles unsuitable for simulation. In contrast, the inclusion of the triangle quality term dramatically improves the minimum angle to 26.57 degrees and concurrently reduces the Hausdorff distance. This clearly demonstrates that the quality term is indispensable for preventing degeneracy and producing a mesh suitable for simulation. Ultimately, our full method, which also incorporates the strain term, achieves an optimization that considers physical properties while maintaining high mesh quality. These results demonstrate that all components introduced in our method are essential for striking a balance among geometric fidelity, physical deformation characteristics, and mesh quality.

6.2. Comparisons with Existing Methods

To evaluate the effectiveness of our method, we first conducted a comparative study with the approach by Dunyach et al. [5], a real-time remeshing method primarily intended for interactive deformation. As shown in Fig. 13, we ap-

plied optimization to the original Hanging Cloth mesh using both methods. Table 4 summarizes the results from the perspectives of mesh quality and geometric fidelity. As this table indicates, a comparison of the Hausdorff distance to the original mesh reveals that our method (0.0675) achieves a smaller value than that of Dunyach et al. (0.0735), numerically confirming that our method can generate a mesh more faithful to the original shape. In particular, as can be seen from the visual comparison in Fig. 13, significant differences from the original shape are observed in the border regions with the method by Dunyach et al., whereas our method resulted in smaller discrepancies. This is because our method employs edge-contraction based on QEM (Quadric Error Metrics) and incorporates corrections for border regions, thereby being designed to accurately preserve border edges. Fig. 14 illustrates the scalar strain field used to drive the optimization in Fig. 13(c) (left) and the set of border edges preserved during QEM contractions (right, highlighted in red).

Furthermore, mesh quality is a crucial aspect. Regarding the Overall Min. Angle, the method by Dunyach et al. generated triangles with very small angles (0.37 degrees), which could compromise simulation stability, whereas our method maintained a relatively high-quality minimum angle of 5.67 degrees. Our method also shows a larger value for the Avg. of Min. Tri. Angles, suggesting higher overall triangle quality. This preservation of triangle quality is critically important as it suppresses the generation of sliver triangles and directly contributes to improving the numerical stability and preventing divergence in elastic body simulations.

Next, to demonstrate the advantage of our method's characteristic of strictly preserving topology, we conducted an evaluation using models with more complex shapes. These models were intentionally chosen as representative examples where topology-agnostic, distance-field-based methods like that of Zheng et al. [35] would likely cause the structure to merge or collapse. Fig. 12 visually illustrates the optimization process of our method applied to the Coral Cuffs and Ornaments models. From these figures, it is evident that our method can adaptively reduce the mesh according to strain and curvature while preserving the original structure, including fine holes and separate parts. When such correspondence is unnecessary, topology-agnostic methods can be preferable (e.g., by filling holes or merging thin parts for more aggressive reduction).

Table 5 presents the quantitative evaluation for these models. For both models, a high reduction rate of over 75% is achieved while keeping the Hausdorff distance low, thus maintaining high fidelity to the original shape. Focusing on mesh quality, the Ornaments model shows the notable result of both the minimum angle and the average of minimum angles improving after optimization. For the Coral Cuffs model, although the minimum angle slightly decreases, the average of minimum angles improves, suggesting that a high-quality mesh suitable for simulation is generated overall. These results demonstrate that our method can effectively optimize even complex shapes by leveraging the important characteristic of topology preservation and considering physical deformation properties and geometric features, thereby achieving an excellent balance between accuracy and element quality.

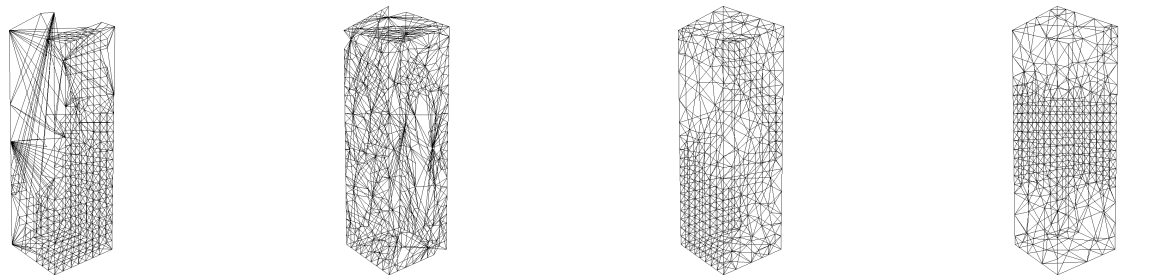
6.3. Limitation and Future Work

This study focuses on *deformation-aware coarsening*: we optimize meshes with QEM-based edge collapses to reduce resolution while preserving elastodynamic fidelity. We intentionally focus on offline coarsening to keep runtime performance stable and the asset pipeline simple. A natural complementary direction is *deformation-guided refinement*, in which resolution is increased where strong elastic response is anticipated (e.g., via edge splits or anisotropic remeshing driven by indicators such as strain energy or modal strain). Exploring such refinement strategies—potentially within a unified coarsen/refine framework—would be an interesting avenue for future work.

Furthermore, although Eq. (6) admits multi-mode aggregation, in this paper we use a single representative low-frequency mode (r=1) to keep the guidance simple and reproducible. Future work includes constrained-mode analysis and multi-condition aggregation to better capture scenario-dependent deformation. This choice can miss scenarios where several deformation mechanisms interact (e.g., bending with twisting or local buckling); in fact, naïvely combining multiple modes may dilute the saliency of the guidance when their hotspots do not coincide. While our framework is designed to be general, fully resolving robust multi-mode selection and weighting within a task-agnostic pipeline is non-trivial. As a pragmatic alternative, task-aware strategies are promising—for example, selecting a small set of complementary modes for cloth draping or fluttering and optimizing/learning their weights w_k in Eq. (6) so that the aggregate field remains spatially selective without dilution.

6.4. Concluding Remarks

To the best of our knowledge, no prior work has integrated modal analysis with topology-preserving QEM-based simplification for elastic-body simulations. Our proposed framework optimizes meshes in accordance with the deformation of elastic bodies using only pre-calculation, reducing computational costs at runtime while suppressing the degradation of visual quality during deformation. This method enhances existing approaches that use a proxy mesh for elastic body simulation. By adopting our optimized mesh as the proxy, we expect to improve both the accuracy



(a) QEM only (b) + Curvature (c) + Triangle quality (d) + Strain (Full score)

Figure 11: Ablation study on the cost function using the Rectangle model. (a) Using only QEM results in a uniform mesh.(b)Adding curvature term. (c) Adding the triangle quality term. (d) Adding the strain term preserves resolution in the center, where deformation is largest.

Table 3: Quantitative results for the ablation study on the Rectangle model.

Cost Function Components	**Hausdorff Dist.(To Original)**	**Min. Angle (°)**	**Avg. Min. Angle (°)**
QEM only	0.125	0.00	37.39
+ Curvature	0.125	0.00	25.04
+ Triangle Quality	0.088	**26.57**	**44.53**
+ Strain (Ours, Full)	0.088	21.80	44.22

Table 4: Comparison of mesh quality metrics and geometric fidelity

Method	**Overall Min. Angle (°)**	**Avg. of Min. Tri. Angles (°)**	**Hausdorff Distance (to Original)**
[5]	0.374822	39.859146	0.073548
Ours	5.678442	42.268261	0.067502

Table 5: Mesh optimization statistics for complex models, comparing geometric fidelity and triangle quality.

Model	**Faces** (Orig. → Opt.)	**Modal Analysis Time (s)**	**Hausdorff Distance**	**Min. Angle (°)** (Orig. → Opt.)	**Avg. Min. Angle (°)** (Orig. → Opt.)
Coral Cuffs	9,998 → 2,500	1.77	0.088	1.40 → 0.42	26.43 → 28.76
Ornaments	5,000 → 1,250	0.60	0.078	0.49 → 1.27	14.21 → 15.50

of collision response and the fidelity of elastic deformation behavior, while maintaining simulation speed. We envision many practical applications in areas such as video games and VR.

We hope that this study and its reference implementation will facilitate further research on the optimization of elastic deformation meshes using modal analysis.

Acknowledgements

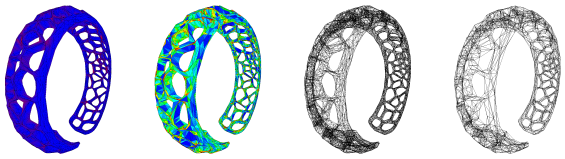
The author thanks Manami Endo for her assistance with the literature survey. The author also acknowledges the support of his employer during his doctoral studies. The author is grateful to the anonymous reviewers for their detailed and constructive comments. Finally, the author thanks his wife and son for their patience and continued support.

References

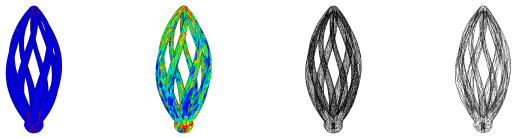
[1] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, page 43–54, New York, NY, USA, 1998. Association for Computing Machinery. 2

[2] J. Barbič and D. L. James. Real-time subspace integration for st. venant-kirchhoff deformable models. *ACM Trans. Graph.*, 24(3):982–990, July 2005. 2, 8

[3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh

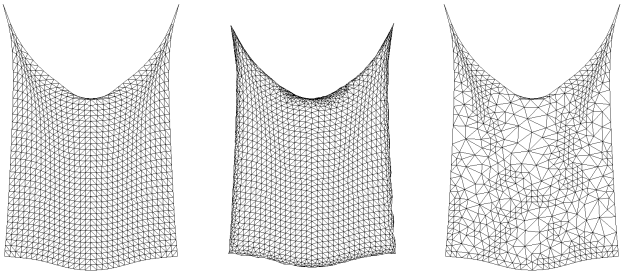


(a) Coral Cuffs



(b) Ornaments

Figure 12: Optimization process for topologically complex models. From left: curvature, strain, original mesh, and optimized mesh.



(a) Original mesh (b) [5] (c) Ours

Figure 13: (a)The original cloth mesh. We will optimize this mesh using both the approach by Dunyach et al.[5](b) and our method(c) for comparison. For the comparison, an unofficial implementation from [6] was used.

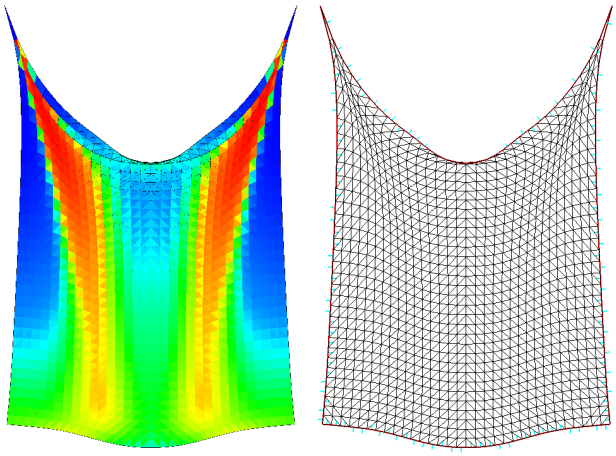


Figure 14: Left: Strain values used in our mesh optimization (Figure 13(c)). Right: Border edges (red line).

Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 9

[4] F. Dassi, A. Mola, and H. Si. Curvature-adapted remeshing of cad surfaces. *Eng. with Comput.*, 34(3):565–576, July 2018. 3

[5] M. Dunyach, D. Vanderhaeghe, L. Barthe, and M. Botsch. Adaptive Remeshing for Real-Time Mesh Deformation. In M.-A. Otaduy and O. Sorkine, editors, *Eurographics 2013 - Short Papers*. The Eurographics Association, 2013. 3, 9, 11, 12

[6] Y. Erel. Implementation of the paper: "adaptive remeshing for real-time mesh deformation". https://github.com/yoterel/adaptive_isotropic_remeshing, 2022. Accessed: June 5, 2025. 12

[7] Z. Ferguson, T. Schneider, D. Kaufman, and D. Panozzo. In-timestep remeshing for contacting elastodynamics. *ACM Trans. Graph.*, 42(4), July 2023. 3, 4

[8] M. Garland. *Quadric-based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University, USA, 1999. AAI9950005. 2, 5, 7

[9] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 209–216, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 1, 2, 5

[10] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. 7

[11] B. Hamann. A data reduction scheme for triangulated surfaces. *Comput. Aided Geom. Des.*, 11(2):197–214, Apr. 1994. 7

[12] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery. 2

[13] H. Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 189–198, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 2

[14] K. Hu, D.-M. Yan, D. Bommes, P. Alliez, and B. Benes. Error-bounded and feature preserving surface remeshing with minimal angle improvement. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2560–2573, 2017. 3

[15] D. Khan, A. Plopski, Y. Fujimoto, M. Kanbara, G. Jabeen, Y. J. Zhang, X. Zhang, and H. Kato. Surface remeshing: A systematic literature review of methods and research directions. *IEEE Transactions on Visualization and Computer Graphics*, 28(3):1680–1713, 2022. 2

[16] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001. 2

[17] M. Macklin, M. Müller, and N. Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, page 49–54, New York, NY, USA, 2016. Association for Computing Machinery. 2

[18] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014. 2

[19] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller. Small steps in physics simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '19, New York, NY, USA, 2019. Association for Computing Machinery. 2, 5, 7, 8

[20] A. Maggiordomo, H. Moreton, and M. Tarini. Micro-mesh construction. *ACM Trans. Graph.*, 42(4), July 2023. 3

[21] G. Marcias, N. Pietroni, D. Panozzo, E. Puppo, and O. Sorkine-Hornung. Animation-aware quadrangulation. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13, page 167–175, Goslar, DEU, 2013. Eurographics Association. 3, 4

[22] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 35–57, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 5

[23] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin. Air meshes for robust collision handling. *ACM Trans. Graph.*, 34(4), July 2015. 7

[24] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, Apr. 2007. 2

[25] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6), Nov. 2012. 3

[26] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, page 215–222, New York, NY, USA, 1989. Association for Computing Machinery. 2

[27] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proceedings of Graphics Interface (GI 1995)*, pages 147–154. Canadian Computer-Human Communications Society, 1995. 2

[28] Y. Qiu, G. Guennebaud, and J. Niesen. Spectra: C++ library for large scale eigenvalue problems. *Retrieved from https://spectralib. org/.*, 2015. 7

[29] B. M. Sarah F. F. Gibson. A survey of deformable modeling in computer graphics. Technical Report TR97-19, MERL - Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, Nov. 1997. 2

[30] S. Sellán, J. Luong, L. Mattos Da Silva, A. Ramakrishnan, Y. Yang, and A. Jacobson. Breaking good: Fracture modes for realtime destruction. *ACM Trans. Graph.*, 42(1), mar 2023. 2, 3, 4

[31] K. Su, N. Lei, W. Chen, L. Cui, H. Si, S. Chen, and X. Gu. Curvature adaptive surface remeshing by sampling normal cycle. *Comput. Aided Des.*, 111(C):1–12, June 2019. 3

[32] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 205–214, New York, NY, USA, 1987. Association for Computing Machinery. 2

[33] J. Wen, J. Barbič, and D. M. Kaufman. Optimal r-adaptive intimestep remeshing for elastodynamics. *ACM Trans. Graph.*, 44(4), July 2025. 3, 4

[34] M. Wicke, D. Ritchie, B. M. Klingner, S. Burke, J. R. Shewchuk, and J. F. O'Brien. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph.*, 29(4), July 2010. 3

[35] Z. Zheng, T. Wang, Q. Feng, Z. Pan, X. Gao, and K. Wu. Proxy asset generation for cloth simulation in games. *ACM Trans. Graph.*, 43(4), July 2024. 1, 3, 4, 7, 10

[36] Q. Zhou and A. Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 8