DOI: 10.1111/cgf.14619
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)

Volume 41 (2022), Number 8

Fast Numerical Coarsening with Local Factorizations

Zhongyun He, Jesús Pérez and Miguel A. Otaduy

Universidad Rey Juan Carlos, Madrid, Spain

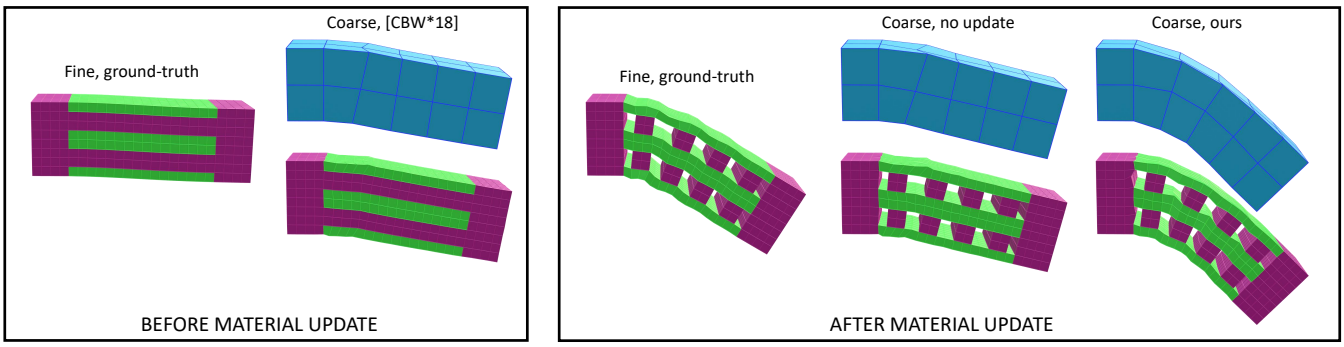

Figure 1: *On the left, we compare a fine-mesh simulation of a heterogeneous material (green is soft, magenta is stiff) vs. the state-of-the-art numerical coarsening method of Chen et al. [CBW*18]. This method is accurate, but it relies heavily on preprocessing. On the right, we update the material dynamically, erasing some stiff cells. State-of-the-art numerical coarsening cannot afford an update of the coarse shape functions, and the simulation is very inaccurate. Our method, on the other hand, allows fast update of the shape functions on-the-fly, and retains the accuracy of state-of-the-art numerical coarsening.*

Abstract

Numerical coarsening methods offer an attractive methodology for fast simulation of objects with high-resolution heterogeneity. However, they rely heavily on preprocessing, and are not suitable when objects undergo dynamic material or topology updates. We present methods that largely accelerate the two main processes of numerical coarsening, namely training data generation and the optimization of coarsening shape functions, and as a result we manage to leverage runtime numerical coarsening under local material updates. To accelerate the generation of training data, we propose a domain-decomposition solver based on substructuring that leverages local factorizations. To accelerate the computation of coarsening shape functions, we propose a decoupled optimization of smoothness and data fitting. We evaluate quantitatively the accuracy and performance of our proposed methods, and we show that they achieve accuracy comparable to the baseline, albeit with speed-ups of orders of magnitude. We also demonstrate our methods on example simulations with local material and topology updates.

CCS Concepts

• ***Computing methodologies*** → ***Physical simulation;***

1. Introduction

In the simulation of heterogeneous objects, the standard approach is to make the resolution of the discretization higher than the resolution of the material heterogeneity, to correctly resolve differences in local deformations. However, when global deformations are the subject of study, and not local deformation details, numerical coarsening offers computational tools to decouple the resolution of the discretization from the resolution of the material heterogeneity. Numerical coarsening uses potentially few degrees of freedom, and interpolates those degrees of freedom in complex nonlinear ways that capture the distribution of deformation of the underlying heterogeneous material. In the context of finite-element elasticity, this is done by estimating rich nonlinear shape functions based on deformation examples [KMOD09, NKJF09, TREO16, CBW*18]. These rich shape functions are evaluated at dense integration points that sample the material heterogeneity, and provide accurate approximation of the integrated elastic response.

Numerical coarsening is a computationally expensive method-

ology; therefore, it is formulated as a preprocess, and at runtime it assumes fixed shape functions. Numerical coarsening loses its charm when material properties change over time, or when objects undergo topology changes, due to the unbearable cost of recomputing shape functions.

In our work, we look at the processes that make numerical coarsening computationally expensive, and we design fast methods for these processes. In particular, we take the state-of-the-art method of Chen et al. [CBW*18] as baseline, which provides the best known accuracy in numerical coarsening in computer graphics. This method decomposes numerical coarsening in two processes: the computation of training data using harmonic displacements [KMOD09], and the estimation of matrix shape functions as a smoothness optimization constrained by the training data.

Our first contribution, described in Section 3, is a fast method for the generation of representative data for numerical coarsening. When a deformable object suffers a local change at runtime, we update only a local factorization of the elasticity problem. Then, we compute deformation examples using a substructuring domain-decomposition method that leverages local factorizations.

Our second contribution, described in Section 4, is a fast method for the optimization of numerical-coarsening shape functions. We decouple optimizing the smoothness of shape functions and matching the training data into two separate optimizations. As a result, we solve multiple decoupled small problems instead of one coupled large problem.

In the paper, we show examples where numerical coarsening becomes practical as a runtime methodology, when material properties change (e.g., due to user actions, strong nonlinearities, topology changes, or activation of smart materials), such as Fig. 1. We also validate quantitatively the accuracy and performance gain of both technical contributions.

2. Related Work

2.1. Numerical Coarsening

The problem of numerical coarsening was first formulated in the context of material homogenization. Given a material with microscale heterogeneity, material homogenization aims to find a mesoscale material model that retains the mesoscale deformation response [ZKO94, PS08]. Coarsening and/or homogenization have been used in the context of various applications in computer graphics, such as design of 3D microstructures for computational fabrication [SBR*15, PZM*15], characterization of 2D microstructures [SMGT18], or thin-shell approximation of yarn-level cloth simulation [SNW20]. While most coarsening methods look at elastostatic deformation response, Chen et al. [CLMK17] propose dynamics-aware coarsening for accurate simulation of dynamic deformations at low resolution.

Kharevych et al. [KMOD09] and Nesme et al. [NKJF09] showed concurrently that, for linear material models, homogenization can be achieved in an optimal way by designing nonlinear shape functions and integrating the material model according to these shape functions.

Then, numerical coarsening can be viewed as the computation of shape functions that capture the distribution of microscale deformation. Regular shape functions produce artificial stiffening of the simulation, as they do not allow the microscale material to deform naturally. Coarsening shape functions, on the other hand, map the coarse displacement field to the fine domain in a complex nonlinear way, and they reproduce effects such as volume preservation or nonlinear deformations induced by heterogeneous material distributions.

One of the research questions in numerical coarsening is to identify coarse deformations that elicit representative microscale deformations to then estimate accurate coarsening functions. The possibilities include Dirichlet boundary conditions on coarse nodes [TREO16], boundary conditions applied on an oversampling region around the coarse element under study [HW97, AB05], or uniform tractions applied on the object's boundary [KMOD09].

An interesting extension of numerical coarsening is to admit matrix shape functions [TREO16], which are key for reproducing volume preservation effects. Chen et al. [CBW*18] decoupled numerical coarsening from material homogenization, and considered arbitrary microscale materials evaluated at dense quadrature points. Due to its generality and demonstrated quality, we use the method of Chen et al. as baseline for our shape function estimation.

The design of rich and expressive shape functions is also addressed from other angles. One is the simulation of arbitrary high-resolution geometry in an accurate way within coarse high-order elements [LLK*20]. Another one is to refine the discretization in a way that is optimal according to the underlying heterogeneity [CBO*19].

2.2. Subspace Simulation

Numerical coarsening can be regarded within the larger family of methods for model order reduction or subspace simulation. These methods search a small number of degrees of freedom and basis functions that accurately represent high-resolution deformations. Some methods build the subspace model from the high-resolution mechanics equations [PW89], but most methods require extensive training data [KLM01]. Recent methods seek nonlinear models synthesize with neural networks, which enable more compact subspaces [FMD*19].

A problem in subspace simulation is to identify integration points, i.e., cubature points, where the material model should be evaluated for accurate results [AKJ08, vTSSH13]. This also reminds the evaluation of the heterogeneous material at high resolution as done by Chen et al. [CBW*18] in numerical coarsening. However, cubature requires very smooth basis functions, whereas coarsening shape functions must be of high frequency.

Another similarity between subspace methods and numerical coarsening is that they rely on computationally expensive preprocessing. Kim and James [KJ09] designed a method to train subspace models at runtime during simulation, in a way to save costly preprocessing, while Yang et al. [YLX*15] introduced fast methods for the generation of training data, the creation of reduced basis functions, and the training of cubature points.

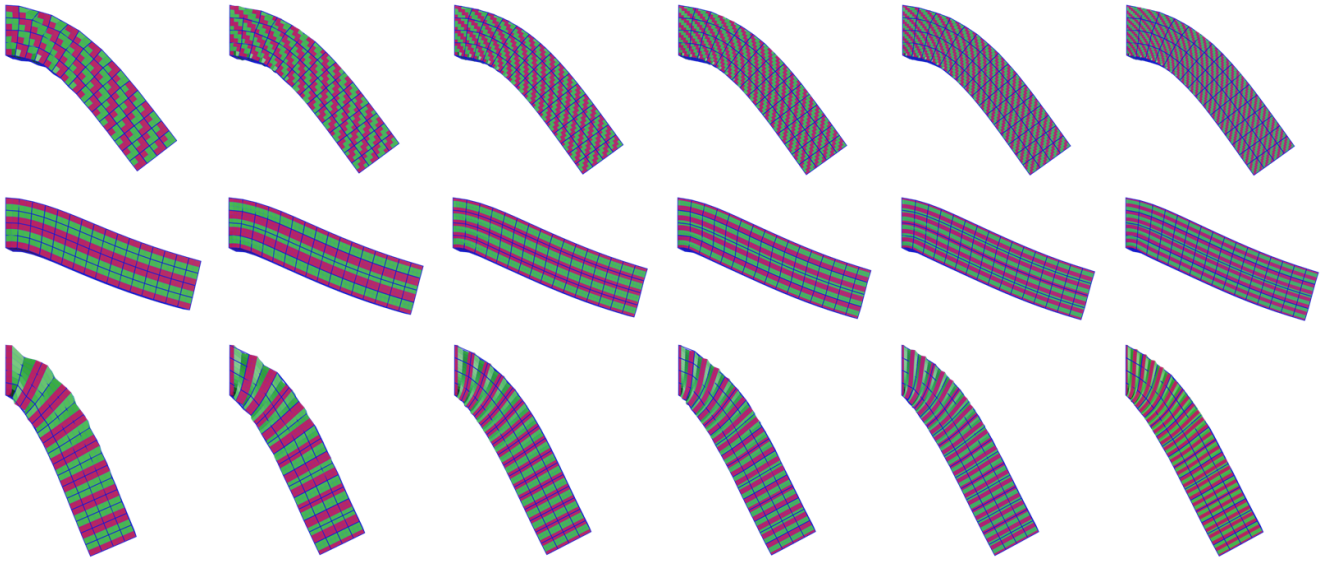

Figure 2: *These images show the test cases for the accuracy of our method of training data generation (see Table 1 for the accuracy evaluation). We have used a beam with three different heterogeneity patterns (oblique, horizontal, vertical), and coarsening degrees ranging from 2 to 7. The stiffness ratio between the stiff material (magenta) and the soft material (green) is 50.*

2.3. Local Factorizations

In our methods, we leverage local Cholesky factorizations within a larger linear solve for the computation of training data. Local factorizations or updates to prefactorizations have often been used to accelerate simulation problems in computer graphics, but we did not find previous methods useful for our target problem.

Hecht et al. [HLSO12] performed local updates to Cholesky factorizations in the context of corotational simulation. By performing these updates only when the error of previous terms was above a threshold, they achieved large net speed-up on the overall simulation. Factorizations can also be updated efficiently under changes to bounday conditions, as demonstrated to produce fast mesh parameterizations [HSH20]. Li et al. [LLKC21] performed local factorization updates in the context of tearing and cutting problems. They used Cholesky factorization in the global pass of a projective dynamics solver, hence the fast factorization updates allowed them to handle simulations with topology changes. On a different direction, Liu et al. [LMAS16] developed a domain-decomposition solver based on Schur complements and local factorizations. Our solver follows a similar approach, but we found no speed-up with standard Schur complement methods, and we tailored a substructuring method to further leverage local factorizations. Finally, Herholz and Alexa [HA18] solved a different but related problem. They factorized the complete mesh of an object, and then they obtained in a fast way factorizations for local submeshes.

3. Training Data with Local Factorizations

The baseline coarsening method uses global deformations as training data. With high-resolution heterogeneous objects, it becomes intractable to compute such global deformations at runtime, when the material or topology of an object changes. We start this section investigating the balance between accuracy and computational speed as a function of the support of training deformations.

Then, we introduce a domain-decomposition solver based on substructuring that leverages local factorizations when material or topology changes are local. Our substructuring solver uses representative subspace constraints between local submeshes, which allows fast solution of the full problem.

3.1. Local Oversampling

In numerical coarsening, nonlinear shape functions approximate the deformation response within each coarse element. Then, the simplest approach to generating training data for each coarse element would only deform the fine submesh overlapping with the coarse element under study. This is actually the approach followed by several numerical coarsening methods [NKJF09, TREO16]. However, this approach is known to produce artifacts [HW97], as the deformations do not account for behaviors induced by material outside the coarse element.

Oversampling [HW97, AB05] refers to the creation of training deformations by applying boundary conditions on a domain larger than the coarse element under study. The question is what size of oversampling region is necessary for the generation of representative training data, without incurring in excessive cost.

Harmonic displacements, introduced by Kharevych et al. [KMOD09], can be regarded as an extreme case of oversampling, with the complete object as deformation domain. Harmonic displacements represent the linearized deformation of the full object under six types of linear tractions applied on

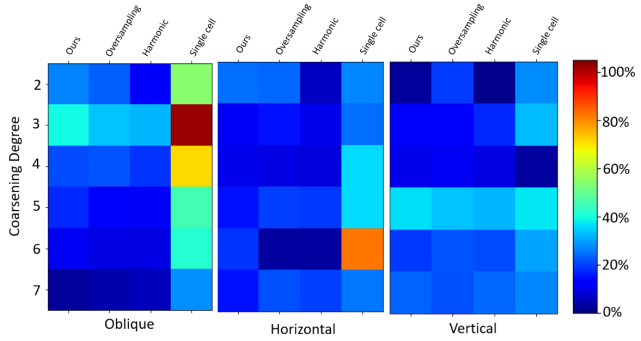

Table 1: *Error of coarsening simulation with respect to a ground-truth fine simulation, under different methods for training data generation. The test uses a beam with the heterogeneity patterns and coarsening degrees shown in Fig. 2. Our substructuring method suffers only a small penalty with respect to complete oversampling or global harmonic displacements.*

Coarse. degree	2	3	4	5	6	7
Oversampling	10.3	92.5	804	3.3e3	11.6e3	28.9e3
Single cell	0.2	0.8	2.0	4.9	11.4	22.9
Ours	1.8	4.9	8.5	16.3	36.4	82.2

Table 2: *Comparison of timings (in milliseconds) for the generation of training data under different methods and coarsening degrees. The timings measure the data generation for one hexahedral cell fully surrounded by 26 cells. Our substructuring method takes about* $3.5\times$ *as long as the single-cell method, but with far superior accuracy as shown in Table 1. The accuracy is close to the oversampling method, but with a speed-up of two orders of magnitude after coarsening degree 4.*

the boundary. While Kharevych et al. used harmonic displacements for the estimation of scalar shape functions, later Chen et al. [CBW*18] successfully used the same target deformations for the estimation of matrix shape functions.

As a trade-off between accuracy and computational cost, we propose to compute harmonic displacements with *one-ring oversampling*, i.e., oversampling with a domain given by the one-ring of the coarse element under study (i.e., 27 coarse hexahedra for interior elements). We have quantified the difference between global harmonic displacements and our one-ring oversampling approach, and we conclude that the difference is minimal. In contrast, single-cell training may suffer excessive deviations.

Fig. 2 shows the examples used in our comparisons, which include 3 different heterogeneity patterns (oblique, horizontal, and vertical), and 6 different coarsening degrees (from 2 to 7), for a hanging beam model with $16\times4\times4$ elements. In Table 1 we report the RMS error (in percentage) of the resulting coarsening simulations with respect to a ground-truth fine simulation, under different conditions for training data generation. Both global harmonic displacements and one-ring oversampling produce accurate results, with harmonic displacements only marginally better. Single-cell training, on the other hand, suffers large errors.

3.2. Substructuring

Under local material updates, we can leverage one-ring oversampling and avoid the large computational cost of global harmonic displacements. However, note that, with coarsening degree d, the number of fine nodes in the one-ring submesh is $(3d+1)^3$. Then, the factorization of the linear system corresponding to the one-ring submesh soon becomes a bottleneck.

We further leverage local updates, and we design a domain-decomposition approach based on substructuring, which reuses local factorizations. Without loss of generaility, we assume that the coarse cell under study is an interior cell, i.e., it has a full one ring with 26 neighboring coarse cells. Substructuring implies formulating an elasticity problem per coarse cell, and using constraints to connect per-cell solutions. We follow a Schur complement approach to first solve Lagrange multipliers for the constraints, and then the actual displacements.

We denote as $\mathbf{u}_i$ a vector collecting all fine-node displacements within each of the 27 coarse cells. Harmonic displacements imply solving linear elasticity problems with Neumann boundary conditions, and we denote as $\mathbf{H}_i$ the local Hessian of each coarse cell, and $\mathbf{f}_i$ its applied external force. Furthermore, we denote each coupling constraint j between pairs of adjacent coarse cells as $\sum_i \mathbf{S}_{j,i}\mathbf{u}_i = 0$. In practice, only two matrices $\mathbf{S}_{j,i}$ are non-zero for each coupling constraint j. For an exact elasticity problem, the $\mathbf{S}_{j,i}$ are selection matrices, with some blocks $\pm\mathbf{I}$ and the rest zero, but in our substructuring approach we approximate the constraints as we will discuss later.

The solution to the one-ring oversampling problem can be expressed as a constrained optimization. With a Lagrange multiplier formulation, this can be written as:

$$\{\mathbf{u}_i,\lambda_j\} = \arg\min \sum_i \frac{1}{2}\mathbf{u}_i^T\mathbf{H}_i\mathbf{u}_i - \mathbf{f}_i^T\mathbf{u}_i + \sum_j \lambda_j^T \sum_i \mathbf{S}_{j,i}\mathbf{u}_i. \quad (1)$$

Following the Schur complement, we first solve for the Lagrange multipliers as:

$$\sum_{i,k} \mathbf{S}_{j,i}\mathbf{H}_i^{-1}\mathbf{S}_{k,i}^T\lambda_k = \sum_i \mathbf{S}_{j,i}\mathbf{H}_i^{-1}\mathbf{f}_i, \quad \forall j. \quad (2)$$

And then we solve for fine displacements as:

$$\mathbf{H}_i\mathbf{u}_i = \sum_k \mathbf{S}_{k,i}^T\lambda_j - \mathbf{f}_i, \quad \forall i. \quad (3)$$

The substructuring solver requires solving many linear systems per coarse cell, with local Hessians $\mathbf{H}_i$, both in (2) and (3). Here, we leverage local factorizations. Without loss of generality, we assume that one coarse cell is modified at a time (i.e., its material is modified, or the cell suffers a topology change). This implies that training data should be updated for this cell and all its 26 neighbors, which requires solving the substructuring problem described above 27 times. However, it is sufficient to update the factorization of the Hessian $\mathbf{H}_i$ of the modified coarse cell. All other Hessians, and hence their factorizations, remain fixed. Then, it is sufficient to update six linear solves in (2), one for each boundary face of the

modified coarse cell. Once the Lagrange multipliers are computed, (3) involves 27 linear solves.

We use Cholesky factorizations to obtain factorizations for the per-cell Hessians $\mathbf{H}_i$. As the Hessians are indefinite, we add a small diagonal regularizer, which does not affect the solution in practice.

In our substructuring solver, assuming full coupling constraints between coarse cells, the bottleneck is the computation of the Lagrange multipliers in (2). There are 54 cell coupling constraints. With coarsening degree d, the number of fine nodes per face of coarse cell is $(d+1)^2$. Then, (2) boils down to a linear system of size $3\times(d+1)^2\times 54$, with all Lagrange multipliers.

We propose a drastic reduction of the cost of (2), by choosing a small set of constraints $\mathbf{S}_{i,j}$ between neighboring coarse cells. We choose two types of constraints:

- Rigid translation constraints. These are formulated by setting the centers of mass of two adjacent cell faces to remain coupled. With two neighboring cells of displacements $\mathbf{u}_i$ and $\mathbf{u}_j$, and centers of mass computed through matrices $\mathbf{s}_i$ and $\mathbf{s}_j$, rigid translation constraints are expressed as $\mathbf{s}_i\mathbf{u}_i-\mathbf{s}_j\mathbf{u}_j=0$. Rigid constraints in substructuring have been used before, e.g., to couple substructuring reduced deformable models [BZ11].
- Harmonic displacement constraints. We wish the substructuring solver to be able to match exactly the global harmonic displacements under the initial material distribution. This ensures that shape function updates converge to the exact solution as dynamic material updates approach zero. To do this, we look at the forces between adjacent cells for each harmonic displacement, and we ensure that a constraint direction is exactly aligned with each of the forces. Given the boundary face forces for all 6 harmonic displacements grouped in a matrix $\mathbf{h}$ (filled with zeros for the rest of the nodes), we express the constraints $\mathbf{h}^T(\mathbf{u}_i-\mathbf{u}_j)=0$.

With rigid translation constraints and harmonic displacement constraints, the linear system in (2) is trimmed to a size of 9×54.

Using the same experiment for comparing oversampling methods discussed in Section 3.1 (see also Fig. 2), we have quantified the accuracy and performance of our efficient substructuring method. As shown in Table 1 and Table 2, the cost of our substructuring solver is just about $3.5\times$ the cost of single-cell data generation, but the accuracy is only slightly worse than for exact one-ring oversampling, albeit more than two orders of magnitude faster.

In all our tests, we have used Eigen [GJ*10] for Cholesky factorizations. We have also investigated the use of iterative solvers, i.e., conjugate gradient. Conjugate gradient is sometimes faster than the Cholesky solver on the one-ring oversampling method, but it becomes slower as the heterogeneity (i.e., ratio of stiffness values) grows, because the system's conditioning becomes worse. Therefore, we chose Cholesky factorization as the robust baseline for comparisons.

4. Local Coarsening Optimization

Given training data for each coarse cell, in this section we describe how we estimate shape functions. Previous work requires solving a large and complex optimization to ensure that shape functions are smooth while fitting the training data. Instead, we decouple these two objectives, and we solve multiple small problems. The resulting shape functions are almost equivalent to the baseline. We start discussing the desiderata of coarsening shape functions, and then we introduce our fast decoupled optimization.

4.1. Shape Functions and their Properties

Let us define notation to walk through the formal definition of coarsening shape functions and their properties. We consider a coarse discretization H and a fine discretization h that overlap on an undeformed domain parameterized by $\mathbf{X}$. In this section, we use subindices i and j to index quantities referring to coarse and fine nodes respectively. Then, $\{\mathbf{X}_i^H\}$ and $\{\mathbf{X}_j^h\}$ denote, respectively, the positions of coarse and fine nodes in undeformed reference space. For the definition of the shape functions, we limit ourselves to one element of the coarse discretization; hence H represents a single coarse element. Within this element, we use matrix shape functions to interpolate the displacement field of coarse nodes.

We denote as $\{\mathbf{M}_{i,j}\}$ the 3×3 matrix shape functions evaluated at fine nodes, for pairs $\mathbf{X}_i^H$ and $\mathbf{X}_j^h$ of coarse and fine nodes. Given displacements evaluated at coarse nodes, $\{\mathbf{u}_i^H\}$, the displacements at fine nodes can be computed as:

$$\mathbf{u}_j^h=\sum_i \mathbf{M}_{i,j}\,\mathbf{u}_i^H. \tag{4}$$

These displacements can be interpolated onto the full fine domain using regular shape functions $\{\mathbf{N}_j(\mathbf{X})\}$ on the fine elements, e.g., trilinear interpolation on hexahedral elements. This yields the following displacement field:

$$\mathbf{u}(\mathbf{X})=\sum_j \mathbf{N}_j(\mathbf{X})\,\mathbf{u}_j^h. \tag{5}$$

Substituting the expression for the displacements of fine nodes (4), we obtain the displacement field as a function of the displacements of coarse nodes:

$$\mathbf{u}(\mathbf{X})=\sum_i \mathbf{N}_i(\mathbf{X})\,\mathbf{u}_i^H,\qquad \text{with } \mathbf{N}_i(\mathbf{X})=\sum_j \mathbf{N}_j(\mathbf{X})\,\mathbf{M}_{i,j}. \tag{6}$$

As a conclusion, we obtain the coarsening shape functions $\{\mathbf{N}_i(\mathbf{X})\}$.

The estimation of shape function coefficients should fulfill several properties: a certain degree of smoothness, geometric invariants that guarantee properties of the deformation, and reproduction of target deformations. We borrow the definition of properties from the state-of-the-art method of Chen et al. [CBW*18].

The smoothness metric $\mathcal{S}$ penalizes the gradient of the coarsening shape functions, and is defined as

$$\mathcal{S}=\sum_i \int_H \operatorname{tr}\left(\nabla\mathbf{N}_i(\mathbf{X})^T:\nabla\mathbf{N}_i(\mathbf{X})\right)d\mathbf{X}. \tag{7}$$

Shape function coefficients should fulfill two geometry constraints: partition of unity

$$\mathcal{G}_{P,j}:\ \sum_i \mathbf{M}_{i,j}-\mathbf{I}=0,\forall j, \tag{8}$$

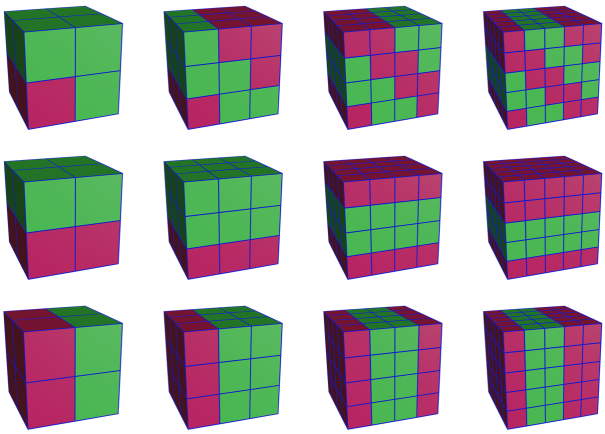

Figure 3: *These images show the test cases for the accuracy of our decoupled optimization of shape functions. They are all single coarse elements with different heterogeneity patterns (oblique, horizontal, vertical), and coarsening degrees ranging from 2 to 5. The stiffness ratio between the stiff material (magenta) and the soft material (green) is 50.*

and rotation invariance

$$\mathcal{G}_{R,j}\colon \sum_i \mathbf{M}_{i,j}\,\text{skew}\left(\mathbf{X}_i^H\right) - \text{skew}\left(\mathbf{X}_j^h\right) = 0, \forall j. \tag{9}$$

skew $(\mathbf{v})$ represents the cross product operation $\mathbf{v}\times$ as a skew symmetric matrix.

Finally, the coarsening shape functions must match the training data, i.e., global harmonic displacements in the work of Chen et al., and one-ring oversampling with substructuring in our case. We denote as $\tilde{\mathbf{u}}_{i,k}^H$ and $\tilde{\mathbf{u}}_{j,k}^h$ coarse and fine node displacements for each training deformation k, computed in our case according to the method described in Section 3.2. Then, training data constraints can be expressed as:

$$\mathcal{D}_{j,k}\colon \sum_i \mathbf{M}_{i,j}\,\tilde{\mathbf{u}}_{i,k}^H - \tilde{\mathbf{u}}_{j,k}^h = 0, \forall j,k. \tag{10}$$

Chen et al. optimize the smoothness $\mathcal{S}$ in (7) subject to geometric constraints (8) and (9) and data constraints (10). This is a large constrained optimization problem per coarse element. With coarsening degree d, $(d+1)^3-8$ fine nodes per coarse element, 72 shape function coefficients per fine node, and 36 constraints per fine node (9 partition-of-unity + 9 rotation-invariance + 6×3 data), it amounts to a problem with $72\times((d+1)^3-8)$ degrees of freedom and $36\times((d+1)^3-8)$ constraints.

4.2. Decoupled Optimization

To describe our decoupled optimization of shape function coefficients, it is convenient to rewrite expressions (7)-(10) in matrix-vector form. To this end, we vectorize all shape function coefficients $\{\mathbf{M}_{i,j}\}$ into a large vector $\mathbf{m}$. This vector includes the coefficients for all fine nodes in the fine discretization h overlapping a coarse element, except for the corner nodes, for which the coefficients are trivially defined by the Kronecker delta.

Coarsening degree	2	3	4	5
[CBW*18]	190	2.6e3	18.9e3	86.1e3
Ours	2.8	5.6	10.7	17.8

Table 3: *Computational cost (in milliseconds) of shape-function optimization for one coarse element, with our decoupled optimization vs. the method of Chen et al. [CBW*18], on several coarsening degrees. We achieve a speed-up of almost two orders of magnitude even on coarsening degree 2, and more than three orders of magnitude on degree 5.*

The smoothness metric (7) is rewritten as

$$\mathcal{S} = \frac{1}{2}\,\mathbf{m}^T\,\mathbf{A}_m\,\mathbf{m} - \mathbf{b}^T\,\mathbf{m}. \tag{11}$$

The geometry constraints (8)-(9) are grouped as

$$\mathcal{G}\colon \mathbf{G}\,\mathbf{m} - \mathbf{g} = 0. \tag{12}$$

And the data constraints (10) are rewritten as

$$\mathcal{D}\colon \mathbf{D}\,\mathbf{m} - \mathbf{d} = 0. \tag{13}$$

To decouple the optimization, we start by separating the effect of constant terms, i.e., the geometry constraints. With a QR decomposition $\mathbf{G}^T = \begin{pmatrix}\mathbf{Q} & \bar{\mathbf{Q}}\end{pmatrix}\begin{pmatrix}\mathbf{R}\\ 0\end{pmatrix}$, the shape function coefficients can be expressed in the null space of the geometry constraints as

$$\mathbf{m} = \bar{\mathbf{Q}}\,\mathbf{z} + \mathbf{m}_0, \tag{14}$$

where $\mathbf{z}$ are the projected shape function coefficients and $\mathbf{m}_0$ is an arbitrary solution to the constraints (e.g., trilinear shape functions). Substituting (14) into (11), we express the smoothness metric subject to geometry constraints as

$$\mathcal{S}_\mathcal{G} = \frac{1}{2}\,(\mathbf{z}-\bar{\mathbf{z}})^T\,\mathbf{A}_z\,(\mathbf{z}-\bar{\mathbf{z}}), \tag{15}$$

$$\text{with } \mathbf{A}_z = \bar{\mathbf{Q}}^T\,\mathbf{A}_m\,\bar{\mathbf{Q}} \text{ and } \bar{\mathbf{z}} = \mathbf{A}_z^{-1}\,\bar{\mathbf{Q}}^T\,(\mathbf{b} - \mathbf{A}_m\,\mathbf{m}_0). \tag{16}$$

Note that $\bar{\mathbf{z}}$ defines optimally smooth shape functions, i.e., ignoring the data constraints.

Given the factorization of the shape functions based on geometry constraints (14), we also express the data constraints (13) subject to the geometry constraints, and we obtain:

$$\mathcal{D}_\mathcal{G}\colon \mathbf{D}\,\bar{\mathbf{Q}}\,\mathbf{z} - \mathbf{d} + \mathbf{D}\,\mathbf{m}_0 = 0. \tag{17}$$

A full optimization of the shape functions would require optimizing $\mathcal{S}_\mathcal{G}$ in (15) subject to the data constraints $\mathcal{D}_\mathcal{G}$ in (17). In contrast, we replace this optimization with a simpler metric but the same goal and constraints. In a nutshell, we minimize the L^2 deviation from the optimally smooth shape functions, subject to the data constraints. Then, we have shape functions defined by:

$$\mathbf{z} = \arg\min_{\mathbf{z}} \frac{1}{2}(\mathbf{z}-\bar{\mathbf{z}})^T\,(\mathbf{z}-\bar{\mathbf{z}}), \quad \text{s.t. } \mathbf{D}\,\bar{\mathbf{Q}}\,\mathbf{z} - \mathbf{d} + \mathbf{D}\,\mathbf{m}_0 = 0. \tag{18}$$

In this optimization, the shape function coefficients for each fine

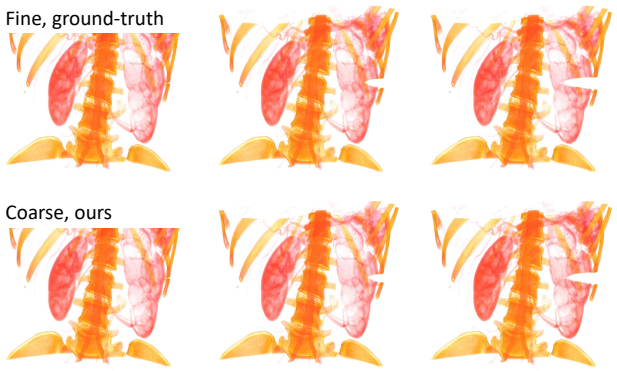

Figure 4: *Simulation of cutting of a CT-scan. By dynamically updating the shape functions of a coarsening model, it is possible to simulate with a coarse mesh the material heterogeneity of a volumetric anatomy model. The images compare the simulation result with a fine mesh and our coarsened model.*

node can be solved completely independently, as the Hessian of the objective is simply identity, and the data constraints are independent for each fine node. We use a Lagrange multiplier method, and first we solve the 18 multipliers per fine node, and then we obtain the shape functions. With coarsening degree d, the cost of computing the shape functions boils down to solving $(d+1)^3 - 8$ linear systems of size 18 each. In spite of solving decoupled optimizations per fine node, global consistency is provided by the globally consistent optimally smooth shape functions and the globally consistent target data.

We have validated the accuracy and performance of our decoupled optimization on the test examples shown in Fig. 3. For these tests, we used the same training data with our method and the method of Chen et al. [CBW*18], computed with harmonic displacements. We have measured the L^2 norm of the difference of the vector of shape function coefficients $\mathbf{m}$, and the error is smaller than 0.001% across all tests. The extreme accuracy of the results indicates that, despite the difference in the optimization metrics in (15) and (18), their gradients are practically aligned at the solution. We cannot prove that this is true for all training data, but we found it to be true for all our test cases. Table 3 summarizes the difference in computational cost per coarse element between our method and the method of Chen et al. [CBW*18]. We achieve a speed-up ranging between almost two and over three orders of magnitude on coarsening degrees between 2 and 5. We did not leverage parallelization in this test, but the performance of our method could be further increased by running per-fine-node computations in parallel.

5. Results

In the examples shown in the paper, we have followed the state-of-the-art method of Chen et al. [CBW*18] for most components except for the dynamic update of coarsening shape functions. At initialization, we compute global harmonic displacements on the full object as training data. This cost is acceptable, as it is executed only once. Using these training data, we optimize coarsening shape functions using our decoupled optimization, as its accuracy is practically comparable to full smoothness optimization. For runtime simulation, we apply the same methodology as Chen et al., which includes the evaluation of the energy, gradient and Hessian using quadrature points on the fine mesh, computation of displacements using a corotational method, and blending of displacements at the boundaries of coarse elements for visualization.

The runtime difference with the state of the art is our dynamic update of coarsening shape functions when the material is modified, which makes the overall approach practical in this setting. The cost of updating numerical coarsening is linear in the number of updated coarse cells, and it includes data generation (Section 3) and optimization of shape functions (Section 4). The runtime update of shape functions can be parallelized in two ways. First, the generation of training data can be parallelized per coarse cell and harmonic displacement. Second, the optimization of shape-function coefficients can be parallelized per fine node. We show two examples of runtime update of the material, to demonstrate applicability of our method. All tests reported in the paper were executed on an Intel Core i7-10750H 2.60GHz x6 CPU with 32 GB of RAM. The timings reported in the paper do not leverage parallelization.

Fig. 1 shows a hanging beam with layers of soft and stiff material. Initially, the response is dominated by the layers of stiff material that run across the beam, and this is well captured by the initial shape functions. During the simulation, we remove fine cells of stiff material, which disconnects the stiff layers, and now the response is dominated by the soft layers. Failing to update the coarsening shape functions leads to a large error of 55% in the final deformation configuration. With our dynamic update of coarsening shape functions, on the other hand, the error is 24%. The beam has a coarsening degree of 4, and we dynamically update the material on all 24 coarse cells. The cost of recomputing shape functions includes 204ms for data generation and 257ms for shape-function optimization. The materials in the beam use a Neo-Hookean model with Young modulus of 10 MPa for the stiff material, Young modulus of 0.1 MPa for the soft material, and Poisson's ratio of 0.45 for both. We compute dynamic simulation using optimization-based backward Euler integration [GSS*15]; please see the video for the comparisons.

Fig. 4 shows a CT volume image that is progressively cut. The volume is represented as a heterogeneous fine mesh with $48 \times 48 \times 36$ elements, at a resolution of 1 element per centimeter. We simulate the deformation on a coarse mesh, then we compute the deformation of the fine mesh nodes, we resample the volume [TREO16], and we volume-render it using Inviwo [JSS*19]. We use a coarsening degree of 2, to ensure that cuts are well represented, although it would be possible to handle coarser meshes with a method that accounts for the topology of both the fine and coarse meshes [NKJF09]. We distinguish two types of materials, soft (Young modulus of 0.1 MPa) and stiff (Young modulus of 100 MPa) based on the opacity of the CT data. Poisson's ratio is 0.45 for both. We advance a blade 1 coarse cell per step, and this requires updating 216 cells per step. The update cost includes 218ms for data generation and 605ms for shape-function optimization. As shown in Fig. 4, the updated coarse mesh matches closely the ground-truth fine simulation. At the final frame, the error is just 28%.

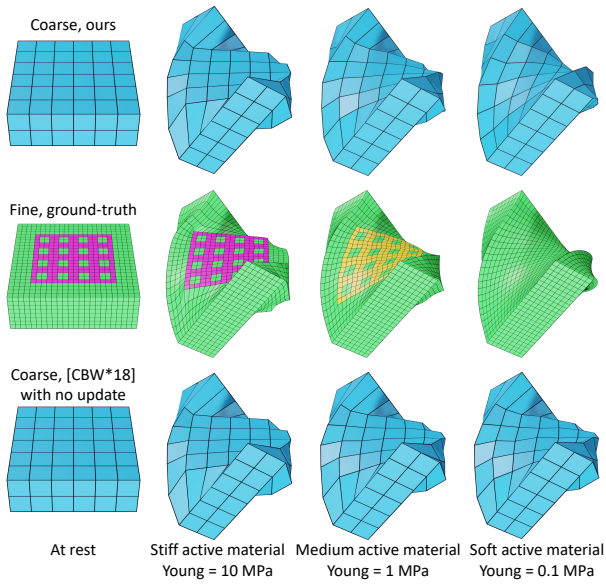

Figure 5: *Twisting a plate with embedded active material. The active material produces a different stiffening behavior of the central part of the plate depending on its regime, which is changed dynamically during the simulation. Our fast and dynamic update of numerical coarsening (top) provides a close match to the fine-mesh behavior (middle) as the active material changes its stiffness. Failing to update coarsening shape functions (bottom) leads to large error due to artificial stiffening of the model.*

As a final example, in Fig. 5 we show dynamic numerical coarsening for a plate filled with active/smart material. The plate is built with a soft substrate (Young modulus = 0.1MPa), in which we have embedded a grid of active material. This material is stiff at rest (Young modulus = 10MPa), producing a reinforcement of the central part of the plate. The active material exhibits two other states, of medium stiffness (Young modulus = 1MPa) and low stiffness (Young modulus = 0.1MPa). Our example motivates applications where these states could be achieved dynamically and progressively through external means, e.g., heating, electrical current, etc. Dynamic modeling of the material behavior is relevant in applications such as soft robot sensing [TKM*20]. In the example, we mesh the plate with a coarse mesh of $6 \times 6 \times 2$ hexahedra, and a fine mesh of $24 \times 24 \times 8$ hexahedra, i.e., coarsening degree = 4. Fig. 5 compares the simulation with the fine mesh (middle row), the coarse mesh with our dynamic update of shape functions (top row), and the coarse mesh with no shape function updates (bottom row). We twist the plate by applying Dirichlet boundary conditions on two opposite sides, showing how our method works well for complex deformations. The coarsening shape functions are initialized with the embedded active material in its stiffest regime. Then, the active material is made progressively softer. Our approach represents accurately the deformations of the fine model, while the no-update approach fails to match the fine deformation when the embedded active material becomes softer. In this case, the inaccuracy of the precomputed shape functions artificially stiffens the plate.

6. Discussion and Future Work

Research in computer graphics keeps exploring different methodologies for building fast simulation models that represent rich, high-resolution detail. Numerical coarsening is one such methodology, and it carries differences with other model-reduction or subspace simulation methods. The major difference is that the basis functions in numerical coarsening remain local, and this has two potential advantages. One advantage is that local support typically allows for more degrees of freedom under the same cost, and hence the ability to resolve deformations of higher resolution. The other potential advantage is to allow for fast local updates. However, in state-of-the-art numerical coarsening methods, the heavy cost of preprocessing prevents local updates, and therefore numerical coarsening methods did not leverage one of their major potential advantages to date.

We have developed a method for accurate dynamic update of numerical coarsening, which allows us to fully leverage the locality of numerical coarsening. The runtime deformation methodology remains the same as in state-of-the-art methods, but shape functions can now be recomputed even at interactive rates under local material updates. We achieve this thanks to technical contributions at both major stages of preprocessing: the generation of training data and the optimization of shape functions.

Our method suffers limitations, which could set directions for future work. Some of the limitations are particular to our work, while others are general disadvantages of numerical coarsening. In particular, the locality of numerical coarsening brings some advantages, but it also limits the ability to represent high-resolution deformations. It might be interesting to increase the support of shape functions, following the oversampling approach, to strike the best balance between accuracy and performance. Another general limitation of numerical coarsening is that the shape functions are built based on a linearized material response, but do not account for nonlinear behaviors. Note that the nonlinear response of the heterogeneous material is well represented, but the shape functions degrade once the nonlinearities are high.

We see two additional extensions that could increase the accuracy of our method, but would also bring performance penalties. One is to use more training data. This implies computing more deformation examples, but possibly also turning the shape-function optimization into a least-squares data-fitting problem. The other one is to increase the number of constraints of the substructuring method, perhaps by adding high-order moments beyond the center of mass.

Acknowledgments. We would like to thank the anonymous reviewers for their feedback. We also want to thank Héctor Barreiro for help with the volume resampling. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 764644. This paper only contains the author's views and the Research Executive Agency and the Commission are not responsible for any use that may be made of the information it contains.

References

[AB05] ALLAIRE G., BRIZZI R.: A multiscale finite element method for numerical homogenization. *Multiscale Modeling & Simulation 4*, 3 (2005), 790–812. 2, 3

[AKJ08] AN S. S., KIM T., JAMES D. L.: Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph. 27*, 5 (2008). 2

[BZ11] BARBIČ J., ZHAO Y.: Real-time large-deformation substructuring. *ACM Trans. Graph. 30*, 4 (2011), 91:1–91:8. 5

[CBO*19] CHEN J., BUDNINSKIY M., OWHADI H., BAO H., HUANG J., DESBRUN M.: Material-adapted refinable basis functions for elasticity simulation. *ACM Trans. Graph. 38*, 6 (2019). 2

[CBW*18] CHEN J., BAO H., WANG T., DESBRUN M., HUANG J.: Numerical coarsening using discontinuous shape functions. *ACM Trans. Graph. 37*, 4 (2018). 1, 2, 4, 5, 6, 7

[CLMK17] CHEN D., LEVIN D. I. W., MATUSIK W., KAUFMAN D. M.: Dynamics-aware numerical coarsening for fabrication design. *ACM Trans. Graph. 36*, 4 (2017). 2

[FMD*19] FULTON L., MODI V., DUVENAUD D., LEVIN D. I. W., JACOBSON A.: Latent-space Dynamics for Reduced Deformable Simulation. *Computer Graphics Forum 38*, 2 (2019), 379–391. 2

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 5

[GSS*15] GAST T. F., SCHROEDER C., STOMAKHIN A., JIANG C., TERAN J. M.: Optimization integrator for large time steps. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 21*, 10 (2015), 1103–1115. 7

[HA18] HERHOLZ P., ALEXA M.: Factor once: Reusing cholesky factorizations on sub-meshes. *ACM Trans. Graph. 37*, 6 (2018). 3

[HLSO12] HECHT F., LEE Y. J., SHEWCHUK J. R., O'BRIEN J. F.: Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans. Graph. 31*, 5 (2012). 3

[HSH20] HERHOLZ P., SORKINE-HORNUNG O.: Sparse cholesky updates for interactive mesh parameterization. *ACM Trans. Graph. 39*, 6 (2020). 3

[HW97] HOU T. Y., WU X.-H.: A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of Computational Physics 134*, 1 (1997), 169 – 189. 2, 3

[JSS*19] JÖNSSON D., STENETEG P., SUNDÉN E., ENGLUND R., KOTTRAVEL S., FALK M., YNNERMAN A., HOTZ I., ROPINSKI T.: Inviwo - a visualization system with usage abstraction levels. *IEEE Transactions on Visualization and Computer Graphics 26*, 11 (2019), 3241–3254. doi:10.1109/TVCG.2019.2920639. 7

[KJ09] KIM T., JAMES D. L.: Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. 28*, 5 (2009), 1–9. 2

[KLM01] KRYSL P., LALL S., MARSDEN J. E.: Dimensional model reduction in non-linear finite element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering 51*, 4 (2001), 479–504. 2

[KMOD09] KHAREVYCH L., MULLEN P., OWHADI H., DESBRUN M.: Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. on Graphics 28*, 3 (2009), 51:1–51:8. 1, 2, 3

[LLK*20] LONGVA A., LÖSCHNER F., KUGELSTADT T., FERNÁNDEZ-FERNÁNDEZ J. A., BENDER J.: Higher-order finite elements for embedded simulation. *ACM Trans. Graph. 39*, 6 (2020). 2

[LLKC21] LI J., LIU T., KAVAN L., CHEN B.: Interactive cutting and tearing in projective dynamics with progressive cholesky updates. *ACM Trans. Graph. 40*, 6 (2021). 3

[LMAS16] LIU H., MITCHELL N., AANJANEYA M., SIFAKIS E.: A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Trans. Graph. 35*, 6 (2016). 3

[NKJF09] NESME M., KRY P. G., JERÁBKOVÁ L., FAURE F.: Preserving topology and elasticity for embedded deformable models. *ACM Trans. on Graphics 28*, 3 (2009), 52:1–52:9. 1, 2, 3, 7

[PS08] PAVLIOTIS G., STUART A.: *Multiscale Methods: Averaging and Homogenization*, vol. 53. 2008. 2

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics 23*, 3 (1989), 215–222. 2

[PZM*15] PANETTA J., ZHOU Q., MALOMO L., PIETRONI N., CIGNONI P., ZORIN D.: Elastic textures for additive fabrication. *ACM Trans. Graph. 34*, 4 (2015), 135:1–135:12. 2

[SBR*15] SCHUMACHER C., BICKEL B., RYS J., MARSCHNER S., DARAIO C., GROSS M.: Microstructures to control elasticity in 3d printing. *ACM Trans. Graph. 34*, 4 (2015), 136:1–136:13. 2

[SMGT18] SCHUMACHER C., MARSCHNER S., GROSS M., THOMASZEWSKI B.: Mechanical characterization of structured sheet materials. *ACM Trans. Graph. 37*, 4 (2018). 2

[SNW20] SPERL G., NARAIN R., WOJTAN C.: Homogenized yarn-level cloth. *ACM Transactions on Graphics (TOG) 39*, 4 (2020). 2

[TKM*20] TAPIA J., KNOOP E., MUTNÝ M., OTADUY M. A., BÄCHER M.: Makesense: Automated sensor design for proprioceptive soft robots. *Soft Robotics 7*, 3 (2020), 332–345. 8

[TREO16] TORRES R., RODRÍGUEZ A., ESPADERO J. M., OTADUY M. A.: High-resolution interaction with corotational coarsening models. *ACM Trans. Graph. 35*, 6 (2016), 211:1–211:11. 1, 2, 3, 7

[vTSSH13] VON TYCOWICZ C., SCHULZ C., SEIDEL H.-P., HILDEBRANDT K.: An efficient construction of reduced deformable objects. *ACM Trans. Graph. 32*, 6 (2013). 2

[YLX*15] YANG Y., LI D., XU W., TIAN Y., ZHENG C.: Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph. 34*, 6 (2015). 2

[ZKO94] ZHIKOV V. V., KOZLOV S. M., OLEINIK O. A.: *Homogenization of differential operators and integral functionals*. Springer-Verlag, 1994. 2