An efficient multigrid method for the simulation of high resolution elastic solids

YONGNING ZHU,
University of California Los Angeles
EFTYCHIOS SIFAKIS, JOSEPH TERAN
University of California Los Angeles – Walt Disney Animation Studios
and ACHI BRANDT
Weizmann Institute of Science

We present a multigrid framework for the simulation of high resolution elastic deformable models, designed to facilitate scalability on shared memory multiprocessors. We incorporate several state-of-the-art techniques from multigrid theory, while adapting them to the specific requirements of graphics and animation applications, such as the ability to handle elaborate geometry and complex boundary conditions. Our method supports simulation of linear elasticity and co-rotational linear elasticity. The efficiency of our solver is practically independent of material parameters, even for near-incompressible materials. We achieve simulation rates as high as 6 frames per second for test models with 256K vertices on an 8-core SMP, and 1.6 frames per second for a 2M vertex object on a 16-core SMP.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Physically based modeling*; G.1.8 [**Numerical Analysis**]: Finite difference methods—*Multigrid and multilevel methods*

Additional Key Words and Phrases: Deformable models, co-rotational linear elasticity, near-incompressible solids, parallel simulation

1. INTRODUCTION

Simulation of deformable solids was introduced to computer graphics by [Terzopoulos et al. 1987; Terzopoulos and Fleischer 1988]. The quest for visual realism has spawned an ever growing interest in simulation techniques capable of accommodating larger, more detailed models. Several researchers have explored approaches such as adaptivity [Debunne et al. 2001; Capell et al. 2002; Grinspun et al. 2002], reduced models [James and Fatahalian 2003; Barbic and James 2005] or shape matching [Müller et al. 2005; Rivers and James 2007] to accelerate the simulation of detailed deformable models, while others used multi-core platforms [Hughes et al. 2007; Thomaszewski et al. 2007] to reduce simulation times.

Multigrid methods [Trottenberg et al. 2001; Brandt 1977] are among the fastest numerical solvers for certain elliptic problems. Due to their efficiency, multigrid methods have garnered attention in the graphics community for a diverse spectrum of applications, including deformable bodies and thin shell simulation [Green et al. 2002; Wu and Tendick 2004; Georgii and Westermann 2006; 2008], mesh deformation [Shi et al. 2006], image editing [Kazhdan and Hoppe 2008], biomedical simulation [Dick et al. 2008], geometry processing [Ni et al. 2004] and in general purpose GPU solvers [Bolz et al. 2003; Goodnight et al. 2003].

We present a scalable framework for fast deformable model simulation based on multigrid techniques. We particularly focus on the simulation of large models with hundreds of thousands of degrees of freedom, and aim to create the ideal conditions for our method to scale favorably on shared memory multiprocessors with a large number of cores. In addition, we accommodate models of arbitrary geometry, and our solver is equally effective even on near-incompressible materials. Although such issues have been individually discussed in the graphics community, we jointly address the challenges of irregular geometry, near-incompressibilty and parallel performance without limiting our scope to smaller problems which can achieve interactive performance with a broader variety of techniques. Our solver accommodates the equations of 3D linear (or co-rotational) elasticity as opposed to simpler elliptic systems (e.g. Poisson problems), and performs well for either dynamic or quasistatic simulation. In order to extract the maximum performance that multigrid formulations have to offer, we adopt some less common choices such as a staggered finite difference discretization and a mixed formulation of the elasticity equations. We believe these decisions are justified by the performance gains, scalability potential and resilience to incompressibility exhibited by our method. *Our main contributions are:*

—We introduce a novel, symmetric boundary discretization, enabling robust treatment of irregular geometry and efficient smoothing of the boundary conditions.

—We show how to accommodate both linear and co-rotational linear elasticity within our framework, for the entire range of compressible to highly incompressible materials.

—We demonstrate the mapping and favorable scalability of our framework on multi-threaded SMP platforms.

2. BACKGROUND

2.1 Linear elasticity

We represent the deformation of an elastic volumetric object using a deformation function $\boldsymbol{\phi}$ which maps any material point $\boldsymbol{X}$ of the undeformed configuration of the object, to its position $\boldsymbol{x}$ in the deformed configuration, i.e. $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X})$. Deformation of an object gives rise to elastic forces [Bonet and Wood 1997] which are analytically given (in divergence form) as $\boldsymbol{f} = \boldsymbol{\nabla}^T \mathbf{P}$ or, componentwise $f_i = \sum_j \partial_j P_{ij}$ where $\mathbf{P}$ is the first Piola-Kirchhoff stress tensor. The stress tensor $\mathbf{P}$ is computed from the deformation map $\boldsymbol{\phi}$. This analytic expression is known as the elastic constitutive equation. We will henceforth adopt the common conventions of using subscripts after a comma to denote partial derivatives, and omit certain summation symbols by implicitly summing over any right-hand side indices that do not appear on the left-hand side of a given

equation. Consequently, the previous equation is compactly written $f_i = P_{ij,j}$. The constitutive equation of linear elasticity is

$$\mathbf{P} = 2\mu\boldsymbol{\epsilon} + \lambda \text{tr}(\boldsymbol{\epsilon})\mathbf{I} \quad \text{or} \quad P_{ij} = 2\mu\epsilon_{ij} + \lambda\epsilon_{kk}\delta_{ij} \tag{1}$$

In this equation, μ and λ are the Lamé parameters of the linear material, and are computed from Young's modulus E (a measure of material stiffness) and Poisson's ratio ν (a measure of material incompressibility) as $\mu = E/(2+2\nu), \lambda = E\nu/((1+\nu)(1-2\nu))$. Also, δ_{ij} is the Kronecker delta, $\boldsymbol{\epsilon}$ is the small strain tensor

$$\boldsymbol{\epsilon} = \tfrac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I} \quad \text{or} \quad \epsilon_{ij} = \tfrac{1}{2}(\phi_{i,j} + \phi_{j,i}) - \delta_{ij} \tag{2}$$

and $\mathbf{F}$ is the deformation gradient tensor, defined as $F_{ij} = \phi_{i,j}$. Using (1,2) we derive the governing equations

$$f_i = \mu\phi_{i,jj} + (\mu + \lambda)\phi_{j,ij} = \mathcal{L}_{ij}\phi_j \tag{3}$$

In this equation $\mathcal{L} = \mu\Delta\mathbf{I} + (\mu + \lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$ is the partial differential operator of linear elasticity. A static elasticity problem amounts to determining the deformation map $\boldsymbol{\phi}$ that leads to an equilibrium of the total forces, i.e. $\mathcal{L}\boldsymbol{\phi} + \boldsymbol{f}_{\text{ext}} = 0$, where $\boldsymbol{f}_{\text{ext}}$ are the external forces applied on the object. For simplicity, we redefine $\boldsymbol{f} = -\boldsymbol{f}_{\text{ext}}$ and the static elasticity problem becomes equivalent to the linear partial differential equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$.

2.2 Multigrid correction scheme

Multigrid methods are based on the concept of a smoother which is a procedure designed to reduce the residual $\boldsymbol{r}=\boldsymbol{f}-\mathcal{L}\boldsymbol{\phi}$ of the differential equation. For example, in a discretized system, Gauss-Seidel or Jacobi iteration are common smoothers. An inherent property of elliptic systems is that when the magnitude of the residual is small, the error $\boldsymbol{e} = \boldsymbol{\phi} - \boldsymbol{\phi}_{\text{exact}}$ is expected to be smooth [Brandt 1986]. Smoothers are typically simple, local and relatively inexpensive routines, which are efficient at reducing large values of the residual (and, as a consequence, eliminating high frequences in the error). Nevertheless, once the high frequency component of the error has been eliminated, subsequent iterations are characterized by rapidly decelerated convergence. Multigrid methods seek to remediate this stagnation by using the smoother as a building block in a multilevel solver that achieves constant rate of convergence, regardless of the prevailing frequencies of the error. This is accomplished by observing that any persistent lower frequency error will appear to be higher frequency if the problem is resampled using a coarser discretization. By transitioning to ever coarser discretizations the smoother retains the ability to make progress towards convergence.

The components of a multigrid solver are:

—Discretizations of the continuous operator $\mathcal{L}$ at a number of different resolutions, denoted as $\mathcal{L}^h, \mathcal{L}^{2h}, \mathcal{L}^{4h}$ etc. (where the superscripts indicate the mesh size for each resolution).

—Smoothing subroutine, defined at each resolution.

—Prolongation and Restriction subroutines. These implement an upsampling and downsampling operation respectively, between two different levels of resolution.

—An exact solver, used for solving the discrete equations at the coarsest level. As the coarse grid is expected to be small, any reasonable solver would be an acceptable option.

Algorithm 1 gives the pseudocode for a V(1,1) cycle of the Multigrid correction scheme, which is the method used in our paper. The following sections provide the implementation details for the components of the multigrid scheme, and explain our design decisions.

Algorithm 1 Multigrid Correction Scheme – V(1,1) Cycle

procedure MULTIGRID($\boldsymbol{\phi}, \boldsymbol{f}, L$) ▷ $\boldsymbol{\phi}$ is the current estimate
  $\boldsymbol{u}^h \leftarrow \boldsymbol{\phi}, \boldsymbol{b}^h \leftarrow \boldsymbol{f}$ ▷ total of $L+1$ levels
  for $l = 0$ **to** $L-1$ **do**
    Smooth($\mathcal{L}^{2^l h}$,$\boldsymbol{u}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
    $\boldsymbol{r}^{2^l h} \leftarrow \boldsymbol{b}^{2^l h} - \mathcal{L}^{2^l h}\boldsymbol{u}^{2^l h}$
    $\boldsymbol{b}^{2^{l+1} h} \leftarrow$Restrict($\boldsymbol{r}^{2^l h}$), $\boldsymbol{u}^{2^{l+1} h} \leftarrow 0$
  end for
  Solve $\boldsymbol{u}^{2^L h} \leftarrow (\mathcal{L}^{2^L h})^{-1}\boldsymbol{b}^{2^L h}$
  for $l = L-1$ **down to** 0 **do**
    $\boldsymbol{u}^{2^l h} \leftarrow \boldsymbol{u}^{2^l h}+$Prolongate($\boldsymbol{u}^{2^{l+1} h}$)
    Smooth($\mathcal{L}^{2^l h}$,$\boldsymbol{u}^{2^l h}$,$\boldsymbol{b}^{2^l h}$)
  end for
  $\boldsymbol{\phi} \leftarrow \boldsymbol{u}^h$
end procedure

3. DISCRETIZATION

Our method uses a staggered finite difference discretization on uniform grids, a familiar practice in the field of computational fluid dynamics (e.g. [Harlow and Welch 1965]). Although far less widespread for the simulation of solids, this formulation was selected for reasons of efficiency and numerical stability.

Use of regular grids. We discretize the elasticity problem on a regular Cartesian lattice. Our deformable model is embedded in this lattice, similar to the approach of [Rivers and James 2007]. Although an unstructured mesh provides more flexibility, we opted for a regular grid for economy of storage. For example, storing the topology of a tetrahedral lattice could easily require 4-5 times more than the storage required for the vertex positions, taking up valuable memory bandwidth. Additionally, the discrete PDE and transfer operators are uniform across regular grids, eliminating the need for explicit storage. Although not used in this paper, adaptivity can also be combined with regular grids, see e.g. [Brandt 1977].

Use of finite differences. Finite elements are arguably the most common discretization method for elasticity applications in graphics (see e.g. [O'Brien and Hodgins 1999]). Finite elements have also been successfully combined with multigrid. However, we based our method on a finite difference discretization for the following reasons: Our method owes its good performance for highly incompressible materials to a mixed formulation of elasticity (section 4.1). Although it is possible to combine this formulation with finite elements (see e.g. [Brezzi and Fortin 1991]) it is much simpler to implement it using finite differences. For regular lattices, both finite elements and finite differences are second-order accurate discretizations away from the boundary, while both are susceptible to degrading to first order near the boundaries as discussed in section 9.2. In addition, our finite difference scheme leads to sparser stencils than finite elements: in our formulation of 3D linear elasticity, each equation has 15 nonzero entries, while 81 entries are required by a trilinear hexahedral finite element discretization, and 45 for BCC tetrahedral finite elements. This translates to a lower computational cost for a finite difference scheme. Finally, as part of our contribution we derive a specific finite difference scheme that guarantees the same symmetry and definiteness properties that are automatic with finite element methods.

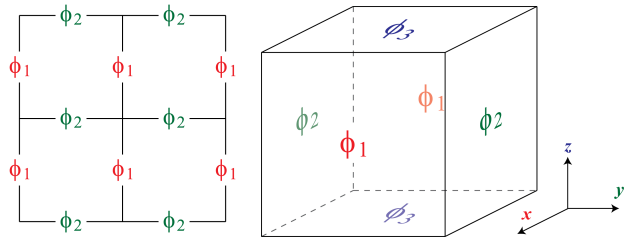

Fig. 1: *Staggering of variables in 2D(left) and 3D(right). Equations $\mathcal{L}_1$,$\mathcal{L}_2$,$\mathcal{L}_3$ are also stored on ϕ_1,ϕ_2,ϕ_3 locations respectively.*

Use of staggered variables. In a regular grid it would be most natural to specify all components of the vector-valued deformation map $\boldsymbol{\phi}$ at the same locations, for example at the nodes of the grid. However, for equation (3) doing so may result in grid-scale oscillations, especially for near-incompressible materials. This is qualitatively analogous to an artifact observed in the simulation of fluids with non-staggered grids, where spurious oscillations may be left over in the pressure field. For multigrid methods, such oscillatory modes are problematic, as they may not respect the fundamental property of elliptic PDEs that a low residual implies a smooth error, requiring more elaborate and expensive smoothers to compensate. We avoid this issue by adopting a staggered discretization (Figure 1), which is free of this oscillatory behavior. More specifically, ϕ_i variables are stored at the centers of grid faces perpendicular to the Cartesian axis vector $\boldsymbol{e}_i$. For example, ϕ_1 values are stored on grid faces perpendicular to $\boldsymbol{e}_1$, i.e. those parallel to the yz-plane. The same strategy is followed in 2D, where faces of grid cells are now identified with grid edges, thus ϕ_1 values are stored at the center of y-oriented edges, and ϕ_2 values at the center of x-oriented edges. We define discrete first-order derivatives using central differences:

$$
\begin{aligned}
D_1 u[x,y,z] &= u[x+\tfrac{1}{2}h_x, y, z] - u[x-\tfrac{1}{2}h_x, y, z] \\
D_2 u[x,y,z] &= u[x, y+\tfrac{1}{2}h_y, z] - u[x, y-\tfrac{1}{2}h_y, z] \\
D_3 u[x,y,z] &= u[x, y, z+\tfrac{1}{2}h_z] - u[x, y, z-\tfrac{1}{2}h_z]
\end{aligned}
$$

where (h_x, h_y, h_z) are the dimensions of the background grid cells. Second-order derivative stencils are defined as the composition of two first-order stencils, i.e. $D_{ij} = D_i D_j$. An implication of these definitions is that the discrete first derivative of a certain quantity will not be collocated with it. For example, all derivatives of the form $D_i\phi_i$ are naturally defined at cell centers, while $D_1\phi_2$ is located at centers of z-oriented edges in 3D, and at grid nodes in 2D. However, derivatives are centered at the appropriate locations for a convenient discretization of (3). In particular, all stencils involved in the discretization of equation $\mathcal{L}_i$ are naturally centered on the location of variable ϕ_i. Thus, the staggering of the unknown variables implies a natural staggering of the discretized differential equations. Figure 2 illustrates this fact in 2D, where the discrete stencils for the operators $\mathcal{L}_1$ and $\mathcal{L}_2$ from (3) are shown to be naturally centered at ϕ_1 and ϕ_2 variable locations, respectively.

4. CONSTRUCTION OF THE SMOOTHING OPERATOR

A majority of elastic materials of interest to computer graphics (e.g. the muscles and flesh of animated characters) are ideally incompressible. A number of authors [Irving et al. 2007; Kaufmann et al. 2008] have discussed the simulation challenges of near-incompressible materials and proposed solutions. For a multigrid

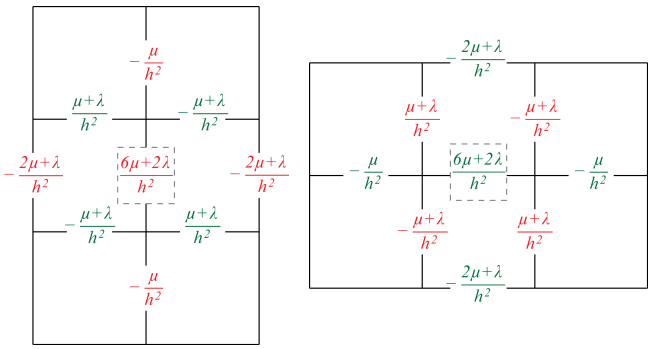

Fig. 2: *Discrete stencils for operators $\mathcal{L}_1$(left) and $\mathcal{L}_2$(right) of the PDE system (3). The red and green nodes of the stencil correspond to ϕ_1 and ϕ_2 values respectively. The dashed square indicates the center of the stencil, where the equation is evaluated.*

solver, naive use of standard smoothers (e.g. Gauss-Seidel) in the presence of high incompressibility could lead to slow convergence or even loss of stability. Our proposed solution is computationally inexpensive and achieves fast convergence independent of material parameters.

4.1 Augmentation and stable discretization

When the Poisson's ratio approaches the incompressible limit $\nu \to 0.5$, the Lamé parameter λ becomes several orders of magnitude larger than μ. As a consequence, the dominant term of the elasticity operator $\mathcal{L} = \mu\Delta\mathbf{I} + (\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$ is the rank deficient operator $(\mu+\lambda)\boldsymbol{\nabla}\boldsymbol{\nabla}^T$; thus $\mathcal{L}$ becomes near-singular. More specifically, we see that any divergence-free field $\boldsymbol{\phi}$ will be in the nullspace of the dominant term, i.e. $\lambda\boldsymbol{\nabla}\boldsymbol{\nabla}^T\boldsymbol{\phi} = 0$. Thus, a solution to the elasticity PDE $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$ could be perturbed by a divergence-free displacement of substantial amplitude, without introducing a large residual for the differential equation. These perturbations can be arbitrarily oscillatory, and lead to high-frequency errors that the multigrid method cannot smooth efficiently or correct using information from a coarser grid. Fortunately, this complication is not a result of inherently problematic material behavior, but rather an artifact of the form of the governing equations. Our solution is to reformulate the PDEs of elasticity into an equivalent system, which does not suffer from the near-singularity of the original. This stable differential description of near-incompressible elasticity is adapted from the theory of mixed variational formulations [Brezzi and Fortin 1991]. We introduce a new auxiliary variable p (which we call *pressure*) defined as $p = -(\lambda/\mu)\boldsymbol{\nabla}^T\boldsymbol{\phi} = -(\lambda/\mu)\mathbf{div}\boldsymbol{\phi}$. We can write

$$
\begin{aligned}
\mathcal{L}\boldsymbol{\phi} &= \mu(\Delta\mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\nabla}^T)\boldsymbol{\phi} + \lambda\boldsymbol{\nabla}(\boldsymbol{\nabla}^T\boldsymbol{\phi}) \\
&= \mu(\Delta\mathbf{I} + \boldsymbol{\nabla}\boldsymbol{\nabla}^T)\boldsymbol{\phi} - \mu\boldsymbol{\nabla}p
\end{aligned} \tag{4}
$$

Thus, the equilibrium equation $\mathcal{L}\boldsymbol{\phi} = \boldsymbol{f}$ is equivalently written as

$$
\begin{pmatrix} \mu(\Delta\mathbf{I}+\boldsymbol{\nabla}\boldsymbol{\nabla}^T) & -\mu\boldsymbol{\nabla} \\ \mu\boldsymbol{\nabla}^T & \frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix} \tag{5}
$$

The top of system (5) follows directly from equation (4), while the bottom is the definition of pressure p. Conversely, the original differential equation (3) can be obtained from (5) by simply eliminating the pressure variable. Thus the augmented differential equation system of (5) is equivalent to the governing equations of linear elasticity (e.g. the two systems agree in the value of $\boldsymbol{\phi}$ when solved).

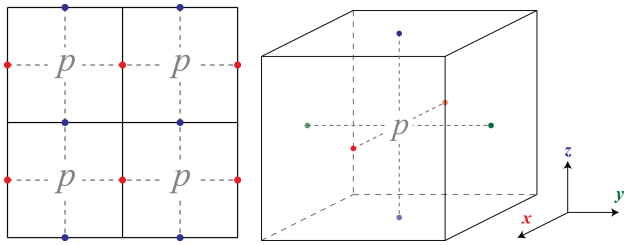

Fig. 3: *Placement of pressures in 2D (left) and 3D (right).*

The important consequence of this manipulation is that this new discretization is stable, in the sense that the system can be smoothed with standard methods without leaving spurious oscillatory modes. This property can be rigorously proved via Fourier analysis; we can verify however that as λ tends to infinity, the term μ^2/λ vanishes, and the resulting limit system is now non-singular. In section 4.2 we describe a simple smoother, specifically tailored to equation (5). The newly introduced pressure variables are also discretized on an offset Cartesian lattice, with pressures stored in cell centers (see Figure 3). Pressure equations are also cell centered. As was the case with the non-augmented elasticity equations, the staggering of deformation ($\boldsymbol{\phi}$) and pressure (p) variables is such that all discrete differential operators are well defined where they are needed.

4.2 Distributive smoothing

The discretization of system (5) yields a symmetric, yet indefinite matrix (discrete first order derivatives are skew-symmetric). Although this system has the stability to admit efficient local smoothing, this cannot be accomplished with a standard Gauss-Seidel or Jacobi iteration. Additionally, for a differential equation such as (5) exhibiting nontrivial coupling between the variables ϕ_1, ϕ_2, ϕ_3 and p, a smoothing scheme which smoothes a given equation by updating several variables at once is often the optimal choice in terms of efficiency [Trottenberg et al. 2001]. The technique we use in our formulation is the distributive smoothing approach. This technique was applied to the Stokes equation in [Brandt and Dinar 1978] while [Gaspar et al. 2008] discussed its application to linear elasticity. In section 7 we generalize it to co-rotated linear elasticity. Let us redefine $\mathcal{L}$ to denote the augmented differential operator of equation (5), and write $\boldsymbol{u} = (\boldsymbol{\phi}, p)$ for the augmented set of unknowns and $\boldsymbol{b} = (\boldsymbol{f}, 0)$ for the right-hand side vector. Thus, system (5) is written as $\mathcal{L}\boldsymbol{u} = \boldsymbol{b}$. Consider the change of variables

$$\begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{I} & -\boldsymbol{\nabla} \\ \boldsymbol{\nabla}^T & -2\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{\psi} \\ q \end{pmatrix} \quad \text{or} \quad \boldsymbol{u} = \mathcal{M}\boldsymbol{v} \qquad (6)$$

where $\boldsymbol{v} = (\boldsymbol{\psi}, q)$ is the vector of auxiliary unknown variables, and $\mathcal{M}$ is called the distribution matrix. In accordance with our staggered formulation, the components ψ_1, ψ_2, ψ_3 of the auxiliary vector $\boldsymbol{\psi}$ will be collocated with ϕ_1, ϕ_2, ϕ_3 respectively, while q and p values are also collocated. Using the change of variables of equation (6), our augmented system $\mathcal{L}\boldsymbol{u} = \boldsymbol{b}$ is equivalently written as $\mathcal{L}\mathcal{M}\boldsymbol{v} = \boldsymbol{b}$. Composing the operators $\mathcal{L}$ and $\mathcal{M}$ yields

$$\mathcal{L}\mathcal{M} = \begin{pmatrix} \mu\Delta\mathbf{I} & 0 \\ \mu(1+\frac{\mu}{\lambda})\boldsymbol{\nabla}^T & -\mu(1+\frac{2\mu}{\lambda})\Delta \end{pmatrix} \qquad (7)$$

That is, the composed system is lower triangular, and its diagonal elements are simply Laplacian operators. This system can be smoothed with any scheme that works for the Poisson equation, including the Gauss-Seidel or Jacobi methods. In fact, the entire system can be smoothed with the efficiency of the Poisson equation, following a forward substitution approach, i.e. we smooth all ψ_1-centered equations across the domain first, followed by sweeps of ψ_2, ψ_3, and q-centered equations in sequence.While we do not necessarily have the auxiliary variables $(\boldsymbol{\psi}, q)$ at our disposal, such an explicit transformation is not necessary. Consider the Gauss-Seidel iteration for the system $\mathcal{L}\boldsymbol{u} = \boldsymbol{b}$: At every step, we calculate a point-wise correction to the variable u_i, such that the residual of the collocated equation $\mathcal{L}_i$ will vanish. That is, we replace variable u_i with $u_i + \delta$ (or $\boldsymbol{u}$ with $\boldsymbol{u} + \delta\boldsymbol{e}_i$) such that:

$$\boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}(\boldsymbol{u} + \delta\boldsymbol{e}_i)) = 0 \Rightarrow (\boldsymbol{e}_i^T\mathcal{L}\boldsymbol{e}_i)\delta = \boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}\boldsymbol{u})$$

The last equation is equivalent to $\mathcal{L}_{ii}\delta = r_i^{\text{old}}$ or $\delta = r_i^{\text{old}}/\mathcal{L}_{ii}$, where $\mathcal{L}_{ii}$ is the i-th diagonal element of the discrete operator and r_i^{old} denotes the i-th component of the residual. In an analogous fashion, a Gauss-Seidel step on the distributed system $\mathcal{L}\mathcal{M}\boldsymbol{v} = \boldsymbol{b}$ amounts to changing ψ_i into $\psi_i + \delta$ (or $\boldsymbol{v}$ into $\boldsymbol{v} + \delta\boldsymbol{e}_i$) such that the i-th residual of the distributed equation is annihilated

$$\boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}\mathcal{M}(\boldsymbol{v} + \delta\boldsymbol{e}_i)) = 0 \Rightarrow \boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}(\boldsymbol{u} + \delta\mathcal{M}\boldsymbol{e}_i)) = 0$$
$$\Rightarrow (\boldsymbol{e}_i^T\mathcal{L}\mathcal{M}\boldsymbol{e}_i)\delta = \boldsymbol{e}_i^T(\boldsymbol{b} - \mathcal{L}\boldsymbol{u}) \Rightarrow \delta = r_i^{\text{old}}/(\mathcal{L}\mathcal{M})_{ii}$$

In this derivation the auxiliary vector $\boldsymbol{v}$ is only used in the form $\mathcal{M}\boldsymbol{v}$ which is equal to the value of the original variable $\boldsymbol{u}$. Thus, after the value of δ has been determined, $\boldsymbol{u}$ is updated to $\boldsymbol{u}+\delta\mathcal{M}\boldsymbol{e}_i$. The computational cost of distributive smoothing is comparable to that of simple Gauss-Seidel iteration, yet it allows efficient smoothing of the equations of linear elasticity, independent of Poisson's ratio. We summarize the distributive smoothing process in Algorithm 2.

Algorithm 2 Distributive Smoothing

```
1: procedure DistributiveSmoothing(L,M,u,b)
2:     for v in {φ1, φ2, φ3, p} do      ▷ Must be in this exact order
3:         for i in Lattice[v] do        ▷ i is an equation index
4:             r ← b_i − L_i · u         ▷ L_i is the i-th row of L
5:             δ ← r/(LM)_ii
6:             u += δ m_i^T              ▷ m_i is the i-th row of M
7:         end for
8:     end for
9: end procedure
```

5. TREATMENT OF BOUNDARIES

The previous sections did not address the effect of boundaries, instead focusing on the treatment of the interior region. The efficiency of the interior smoother can be evaluated using a periodic domain. In fact, it is known [Brandt 1994] that a boundary value problem can be solved at the same efficiency as a periodic problem, at the expense of more intensive smoothing at the boundary. In theoretical studies, the computational overhead of this additional boundary smoothing is often overlooked, as the cost of interior smoothing is *asymptotically* expected to dominate. Nevertheless, practical problem sizes may never reach the asymptotic regime and slow, generic boundary smoothers can pose a performance bottleneck. In this section, we develop a boundary discretization strategy, including a novel treatment of traction boundary conditions, that facilitates the design of efficient and inexpensive boundary smoothers.

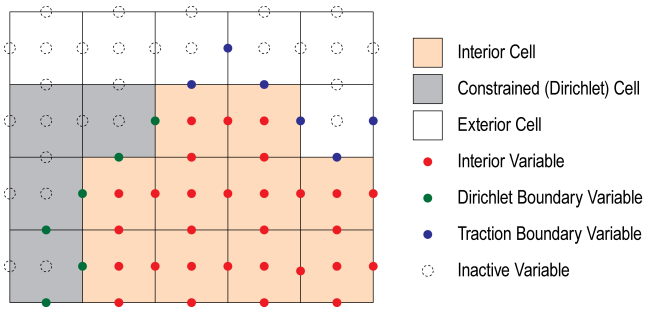

Fig. 4: *Classification of cells,variables and equations near the boundary.*

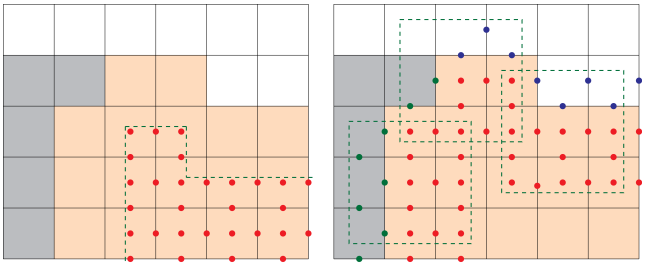

Fig. 5: *Left: Extent of distributive smoothing (interior region), Right: Boundary region with some boxes used by the box smoother.*

5.1 Domain description

Our geometrical description of the computational domain is based on a partitioning of the cells of the background grid (Figure 4). Initially, cells that have an overlap with the simulated deformable body are characterized as *interior* cells, otherwise they are designated *exterior* cells. Additionally, any cell can be user-specified to be a constrained (or *Dirichlet*) cell, overriding any interior/exterior designation this cell may otherwise carry. Dirichlet cells practically correspond to kinematically constrained parts of the object. This classification into interior, exterior and Dirichlet cell types provides an intuitive way to specify the degrees of freedom of our problem, and define their associated equations. We categorize discrete variables and equations as follows:

—*Interior variables and equations*. Any of the variables ϕ_1, ϕ_2, ϕ_3 or p located strictly inside the interior region (i.e. either on an interior cell center, or on the face between two interior cells) is designated an interior variable. For every interior variable, we label its collocated equation from (5) as an *interior equation* and we include this equation as part of our discrete system. Locations of interior variables and equations are depicted as red dots in figure 4.

—*Boundary variables and equations*. Certain interior equations (near the boundary) have a discrete stencil that extends onto variables that are not interior variables themselves. We label these as *boudary variables*. More specifically, a boundary variable is designated a Dirichlet boundary variable if it touches a Dirichlet cell (either inside or on the boundary of one), otherwise it is designated a traction boundary variable. Dirichlet and traction boundary variables are depicted as green and blue dots respectively, in figure 4. Similar to interior variables, for every boundary variable we add a boundary equation (or *boundary condition*) to our discrete system, in order to have the same number of equations as unknowns. Dirichlet variables are matched with a boundary condition of the form $\boldsymbol{\phi}(\boldsymbol{X}) = \boldsymbol{c}$, while traction variables are associated with a condition of the form $\mathbf{P}(\boldsymbol{X})\boldsymbol{N} = \boldsymbol{t}$, where $\boldsymbol{N}$ is the surface normal ($\boldsymbol{t} = 0$ corresponds to a free boundary).

—Variables that have not been designated interior or boundary are labeled *inactive* and can be ignored (depicted as dashed circles in figure 4). No equation is added to our system for these inactive grid locations.

5.2 A general-purpose box smoother

Although a well-posed system can be constructed as described, the distributive smoothing scheme is not valid near the boundary, as the distribution extends outside the domain. In such situations a box smoother [Brandt and Dinar 1978] is a broadly applicable solution. This process amounts to collectively solving a number of equations in a subdomain, simultaneously adjusting the values of all variables within. Our complete smoothing procedure starts with a boundary box smoothing sweep, proceeds with interior distributive smoothing and finishes with a last boundary pass. An interior equation is smoothed distributively if the stencil of its respective equation in the composed system $(\mathcal{LM})\boldsymbol{v} = \boldsymbol{b}$ only includes interior variables, as illustrated in Figure 5 (left). For the boundary, we use overlapping boxes that are two grid cells wide, and centered at the outermost layer of interior cells, as seen in Figure 5 (right). In our experiments the box smoother performed very well, generally achieving near-optimal efficiency for the entire multigrid scheme. In practice, however, this good convergence behavior came at the cost of an enormous computational overhead. This added cost stems from the need to solve a coupled linear system within each box. The computational effort spent on boundary smoothing was often two orders of magnitude more than the cost of interior smoothing on models with tens of thousands of vertices; although the interior cost scales with volume and the boundary cost scales with surface area, even with million-vertex models the cost of the boundary smoother would still dominate by a factor of 10. We address this issue in the next section by designing an effective, yet simple and inexpensive boundary smoother.

5.3 A fast symmetric Gauss-Seidel smoother

We propose a novel formulation that enables equation-by-equation smoothing that is both efficient and inexpensive. The main obstacle to efficient equation-by-equation boundary smoothing schemes, is lack of symmetry, definiteness or diagonal dominance. Additionally, discretizations of the boundary conditions (especially traction) can easily result in loss of symmetry. An alternative local smoother is the Kaczmarz method [Trottenberg et al. 2001], which does not require symmetry or definiteness; we have nevertheless found it to converge extremely slowly, requiring a very large number of iterations. Our solution stems from a new perspective of the constitutive equations and the boundary conditions.

We will show that it is possible to construct a symmetric negative definite discretization that uses only interior variables (as defined in section 5.1). First, we revisit the constitutive equation of linear elasticity (1). The scalar coefficient $\mathrm{tr}(\boldsymbol{\epsilon})$ appearing in equation (2) is equivalently written as $\mathrm{tr}(\boldsymbol{\epsilon}) = \sum_i \epsilon_{ii} = \sum_i \phi_{i,i} - d$, where $d = \mathrm{tr}(\mathbf{I})$ equals the number of spatial dimensions. Similarly, the last equation of system (5) is equivalent to $-(\mu/\lambda)p = \boldsymbol{\nabla}^T\boldsymbol{\phi} = \sum_i \phi_{i,i}$.

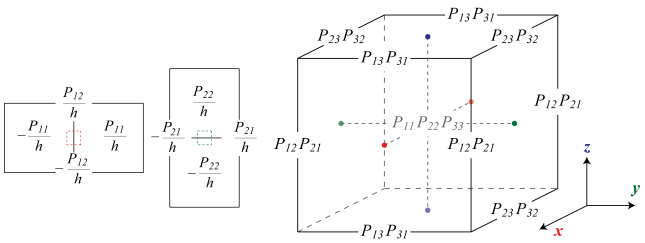

Fig. 6: *Left: Equations $\mathcal{L}_1$,$\mathcal{L}_2$ expressed as divergence stencils. Right: Placement of the components of stress tensor $\mathbf{P}$ in 3D.*

Thus, we have $\text{tr}(\boldsymbol{\epsilon}) = -(\mu/\lambda)p - d$, and equation (1) becomes

$$\mathbf{P} = \mu(\mathbf{F} + \mathbf{F}^T) - \mu p\mathbf{I} - (2\mu + d\lambda)\mathbf{I} \tag{8}$$

The difference between equations (1) and (8) is that the original definition of stress is physically valid for any given deformation field ϕ while the formulation of equation (8) will correspond to the real value of stress only when the augmented system (5) is solved exactly. In detail, the diagonal and off-diagonal components of the stress tensor $\mathbf{P}$ are given as:

$$\begin{aligned} P_{ii} &= 2\mu\phi_{i,i} - \mu p - (2\mu + d\lambda) \\ P_{ij} &= \mu(\phi_{i,j} + \phi_{j,i}). \qquad (i \neq j) \end{aligned}$$

The finite difference approximations of these stress values are:

$$\begin{aligned} P_{ii}(\mathbf{X}) &= 2\mu\frac{\phi_i(\mathbf{X}+\frac{h_i}{2}\boldsymbol{e}_i) - \phi_i(\mathbf{X}-\frac{h_i}{2}\boldsymbol{e}_i)}{h_i} \\ &\quad -\mu p(\mathbf{X}) - (2\mu + d\lambda) \end{aligned} \tag{9}$$

$$\begin{aligned} P_{ij}(\mathbf{X}) &= \mu\left[\frac{\phi_i(\mathbf{X}+\frac{h_j}{2}\boldsymbol{e}_j) - \phi_i(\mathbf{X}-\frac{h_j}{2}\boldsymbol{e}_j)}{h_j}\right. \\ &\quad \left. + \frac{\phi_j(\mathbf{X}+\frac{h_i}{2}\boldsymbol{e}_i) - \phi_j(\mathbf{X}-\frac{h_i}{2}\boldsymbol{e}_i)}{h_i}\right] \end{aligned} \tag{10}$$

The staggering of the position and pressure variables implies a natural placement of the stress values, in accordance with equations (9) and (10). Diagonal stress components are always located at cell centers, while off diagonal components $P_{ij}(i \neq j)$ are node-centered in 2D and edge-centered in 3D (see Figure 6, right). In a fashion similar to our classification of variables and equations, we label stresses as *interior* if their discrete stencils (defined in equations (9) and (10)) contain only interior or Dirichlet variables, while *boundary* stresses include at least one traction boundary variable in their stencil. Interior and boundary stresses are depicted in green and red, respectively, in figure 7 (left).

We can verify that the *position* equations $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ of system (5) are equivalent to the divergence form $\mathcal{L}_i\boldsymbol{u}=\partial_j P_{ij}$, where $\mathbf{P}$ is now given by the new definition of equation (8). The discrete stencils for these equations can be constructed as a two-step process. First, we construct a finite difference discretization for each equation $f_i = \partial_j P_{ij}$, treating every value P_{ij} appearing in this stencil as a separate variable:

$$f_i(\mathbf{X}) = \sum_{j=1}^{d} \frac{P_{ij}(\mathbf{X}+\frac{h_j}{2}\boldsymbol{e}_j) - P_{ij}(\mathbf{X}-\frac{h_j}{2}\boldsymbol{e}_j)}{h_j} \tag{11}$$

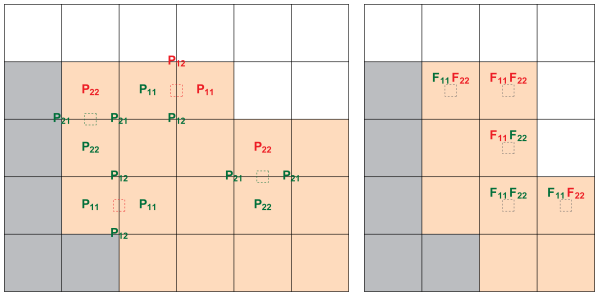

Fig. 7: *Left: Stress variables used in the divergence form of certain interior equations. Boundary stress variables are colored red, interior stresses are green. All boundary stresses can be set to a specific value using a traction condition from a nearby boundary. Right: Interior and boundary gradients used in pressure equations.*

See figure 6 (left) for a visual illustration of this divergence stencil in 2D. As a second step, we replace the stress values P_{ij} in equation (11) using either a finite difference approximation, or a boundary condition. Each of the stress values P_{ij} (4 stresses in 2D, 6 in 3D) can either be an interior or a boundary stress. For all interior stresses, we simply substitute the appropriate finite difference stencil, from equation (9) or (10). For boundary stresses, instead of computing them using a finite difference, we assume that their value is *known* by virtue of a traction boundary condition, thus this value can be simply substituted in equation (11). The assumption that every boundary stress is determined by a traction boundary condition is justified as follows:

—Stress variables of the form $\mathbf{P}_{ij}(i \neq j)$ are centered on grid edges in 3D (see Figure 6, right) and on grid nodes in 2D. This stress variable appears in the finite difference approximation of the term $\partial_j P_{ij}$ in the interior equation $\mathcal{L}_i$. Let $\boldsymbol{X}^*$ be the location where equation $\mathcal{L}_i$ is centered. The stress variable P_{ij} is located one half of a grid cell away from $\boldsymbol{X}^*$, *along the direction* $\boldsymbol{e}_j$. Without loss of generality, assume P_{ij} is located at $\boldsymbol{X}^*+\frac{h_j}{2}\boldsymbol{e}_j$. P_{ij} neighbors exactly four cells; out of those, the two centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i$ are *interior* cells, since we assumed that $\mathcal{L}_i$ was an interior equation. The two other neighbor cells of P_{ij} are centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i + h_j\boldsymbol{e}_j$. We can verify that if those two cells were interior or Dirichlet, P_{ij} would have been an *interior* stress. Thus, P_{ij} is a boundary stress and one of the cells centered at $\boldsymbol{X}^* \pm \frac{h_i}{2}\boldsymbol{e}_i + h_j\boldsymbol{e}_j$ must be exterior. This means that P_{ij} is incident on a traction boundary face perpendicular to the direction $\boldsymbol{e}_j$, and there exists a traction condition $\mathbf{P}\boldsymbol{e}_j = \boldsymbol{t}$ that specifies a value $P_{ij} = t_i$ for this component of the stress. For a free boundary we simply have $P_{ij} = 0$.

—Stress variables of the form $\mathbf{P}_{ii}$ are located at cell centers, and appear in the finite difference approximation of $\partial_i P_{ii}$ in the interior equation $\mathcal{L}_i$. Similar to the previous case, P_{ii} is located one half grid away from the location $\boldsymbol{X}^*$ of $\mathcal{L}_i$ *along the direction* $\boldsymbol{e}_i$. Without loss of generality, assume P_{ii} is located at $\boldsymbol{X}^*+\frac{h_i}{2}\boldsymbol{e}_i$. From (9) we see that the stencil for P_{ii} contains the variables $\phi_i(\boldsymbol{X}^*), p(\boldsymbol{X}^*+\frac{h_i}{2}\boldsymbol{e}_i)$ which are both *interior* (since $\mathcal{L}_i$ is interior) and the one additional variable $\phi_i(\boldsymbol{X}^*+h_i\boldsymbol{e}_i)$. P_{ii} would be a boundary stress only if $\phi_i(\boldsymbol{X}^*+h_i\boldsymbol{e}_i)$ was an exterior variable; in this case P_{ii} would have been "near" (specifically half a cell away from) a traction boundary face normal to $\boldsymbol{e}_i$. Once again, we will use the traction condition associated with this boundary

to set $P_{ii} = t_i$ (or $P_{ii} = 0$ for a free boundary). The subtlety of this formulation is that the stress variable P_{ii} is not located exactly on the boundary; nevertheless the discrete stencil for P_{ii} is still a valid *first-order* approximation of the P_{ii} *at* the boundary.

In summary, we have justified that all *boundary* stress variables can be eliminated (and replaced with known constants) from the divergence form of interior position equations. Notably, for equations that are far enough from the traction boundary (specifically, those that do not require any boundary stresses in equation (8)), this process yields exactly the same results as the direct discretization of system (5). A similar treatment is performed on the discretization of the pressure equation $\mathcal{L}_p{=}\mu \sum_i F_{ii} + \frac{\mu^2}{\lambda} p$. Similar to stresses, the deformation gradients F_{ii} are also characterized as interior or boundary, based on whether they touch traction boundary variables. Since $P_{ii} = 2\mu F_{ii} - \mu p - (2\lambda + d\mu)$, we observe that F_{ii} is boundary if and only if the stress P_{ii} is boundary (see figure 7, right). For such boundary gradients or stresses we can use the traction condition $P_{ii} = t_i$ to eliminate F_{ii} from the pressure equation. This is accomplished by replacing $\mathcal{L}_p \leftarrow \mathcal{L}_p - \frac{1}{2}(P_{ii} - t_i)$ for every boundary gradient F_{ii}.

Our manipulations effectively remove all traction boundary variables from the discretization of the interior equations. For every Dirichlet boundary variable, we assume a Dirichlet condition of the form $\phi_i = c_i$ is provided. Thus, we can substitute a given value for every Dirichlet variable in the stencil of every interior equation that uses it. As a result, our overall discrete system can be written as $\mathcal{L}^* \boldsymbol{u}^* = \boldsymbol{b} - \boldsymbol{b}^D = \boldsymbol{b}^*$, where $\boldsymbol{u}^*$ only contains interior variables, and $\boldsymbol{b}^D$ results from moving the Dirichlet conditions to the right-hand side. The discrete system matrix $\mathcal{L}^*$ has as many rows and columns as interior variables, and will differ from $\mathcal{L}$ near the boundaries, as it incorporates the effect of the boundary conditions. An analysis of our formulation can verify that $\mathcal{L}^*$ has the form

$$\mathcal{L}^* = \begin{pmatrix} \mathbf{L}_\phi & \mathbf{G} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix}$$

In this formulation $\mathbf{L}_\phi$ is symmetric, negative definite, and $\mathbf{D}_p$ is a diagonal matrix with positive diagonal elements. As a final step, we define the *substitution* matrix $\mathcal{U}$

$$\mathcal{U} = \begin{pmatrix} \mathbf{I} & -\mathbf{G}\mathbf{D}_p^{-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

and use it to pre-multiply our equation as

$$\mathcal{U}\mathcal{L}^*\boldsymbol{u}^* = \begin{pmatrix} \mathbf{L}_\phi - \mathbf{G}\mathbf{D}_p^{-1}\mathbf{G}^T & \mathbf{0} \\ \mathbf{G}^T & \mathbf{D}_p \end{pmatrix} \boldsymbol{u}^* = \mathcal{U}\boldsymbol{b}^* \qquad (12)$$

The top left block $\mathbf{L}_\phi - \mathbf{G}\mathbf{D}_p^{-1}\mathbf{G}^T$ is essentially a symmetric and negative definite discretization of our non-augmented system (3) and can be smoothed via Gauss-Seidel iteration. The boundary and interior regions are smoothed in separate sweeps; during the sweep of the boundary smoother, all interior variables not being smoothed are effectively treated as Dirichlet values. The boundary smoother is confined in a narrow region between boundary conditions (variables of this narrow band are depicted in red in Figure 5, right). This narrow support of the boundary smoother has a strong stabilizing effect, and compensates for any difficulties encountered with near-incompressible materials. In practice, we found that 2 Gauss-Seidel boundary sweeps for every sweep of the distributive interior smoother are sufficient for Poisson's ratio up to $\nu = .45$, while 3-4 Gauss-Seidel sweeps suffice for values as high as $\nu = .495$. Finally, we note that Gauss-Seidel is not the only option for smoothing the discrete system derived in this section; in fact it is even possible to use a distributive smoother as in Algorithm 2, taking care to restrict the distribution stencil to active variables.

After completing the smoothing process, we need to update the values of the pressure and traction boundary variables that were previously annihilated. Since the lower right block of equation (12) is diagonal, all pressure equations can be satisfied exactly via a simple Gauss-Seidel sweep. Similarly, the boundary traction variables can be updated using the traction conditions $P_{ij} = t_i$ in a simple back-substitution step, first updating variables on faces between interior and exterior cells, and then variables located at half-cell distance away from the interior region. Notably, at the end of the process all boundary conditions are satisfied *exactly* (i.e. they will have zero residuals), which simplifies our inter-grid transfers discussed next.

6. CONSTRUCTION OF THE TRANSFER OPERATORS

We designed the Restriction ($\mathcal{R}$) and Prolongation ($\mathcal{P}$) operators employed by algorithm 1 aiming to keep implementation as inexpensive as possible, while conforming to the textbook accuracy requirements for full multigrid efficiency (see [Trottenberg et al. 2001]). We define the following 1D averaging operators

$$\mathcal{B}^1 u[x] = \tfrac{1}{2}u[x{-}\tfrac{h}{2}] + \tfrac{1}{2}u[x{+}\tfrac{h}{2}]$$
$$\mathcal{B}^2 u[x] = \tfrac{1}{4}u[x{-}h] + \tfrac{1}{2}u[x] + \tfrac{1}{4}u[x{+}h]$$
$$\mathcal{B}^3 u[x] = \tfrac{1}{8}u[x{-}\tfrac{3h}{2}] + \tfrac{3}{8}u[x{+}\tfrac{h}{2}] + \tfrac{3}{8}u[x{-}\tfrac{h}{2}] + \tfrac{1}{8}u[x{+}\tfrac{3h}{2}]$$

The restriction and prolongation operators will be defined as tensor product stencils of the preceding 1D operators as

$$\mathcal{R}_1 = \mathcal{B}^2 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1 \qquad \mathcal{P}_1^T = 8\,\mathcal{B}^2 \otimes \mathcal{B}^3 \otimes \mathcal{B}^3$$
$$\mathcal{R}_2 = \mathcal{B}^1 \otimes \mathcal{B}^2 \otimes \mathcal{B}^1 \qquad \mathcal{P}_2^T = 8\,\mathcal{B}^3 \otimes \mathcal{B}^2 \otimes \mathcal{B}^3$$
$$\mathcal{R}_3 = \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^2 \qquad \mathcal{P}_3^T = 8\,\mathcal{B}^3 \otimes \mathcal{B}^3 \otimes \mathcal{B}^2$$
$$\mathcal{R}_p = \mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1 \qquad \mathcal{P}_p^T = 8\,\mathcal{B}^1 \otimes \mathcal{B}^1 \otimes \mathcal{B}^1$$

where $\mathcal{R}_i$,$\mathcal{P}_i$ are the restriction and prolongation operators used for variable $\boldsymbol{u}_i$, respectively, and $\mathcal{R}_p$,$\mathcal{P}_p$ are the operators used for the pressure variables. Our restriction and prolongation are not the transpose of one another (as commonly done in other methods) but this practice is not unusual or problematic, see e.g. [Brandt 1977]. Our domain description for the finest grid was based on a partitioning of the cells into interior, exterior and Dirichlet. The coarse grid is derived from the natural 8-to-1 coarsening of the Cartesian background lattice. Furthermore, a coarse cell is designated a Dirichlet cell if *any* of its eight fine sub-cells is Dirichlet. If any of the fine sub-cells are interior and none is Dirichlet, the coarse cell will be considered interior. Otherwise, the coarse cell is exterior. Thus, the coarse active domain is geometrically a superset of the fine, while its Dirichlet parts are extended. Despite this geometrical discrepancy, which is no larger than the grid cell size, we were still able to obtain a highly efficient multigrid scheme as described next.

In our treatment of boundary conditions in section 5.3 we effectively forced all boundary conditions to be satisfied *exactly* after every application of the smoother. In general, if a smoother leaves a residual on the boundary conditions, this residual has to be restricted. In our case all boundary residuals in the fine grid are zero, thus all coarse boundary conditions will be *homogeneous*; for Dirichlet equations they will have the form $u_i^{2h} = 0$ (i.e. the coarse grid incurs no correction), while traction equations will be of the form $\hat{P}_{ij}^{2h} = 0$, where $\hat{\mathbf{P}} = \mu(\mathbf{F} + \mathbf{F}^T) - \mu p \mathbf{I}$ is the homogeneous part of $\mathbf{P}$. We also note that, due to the possible ge-

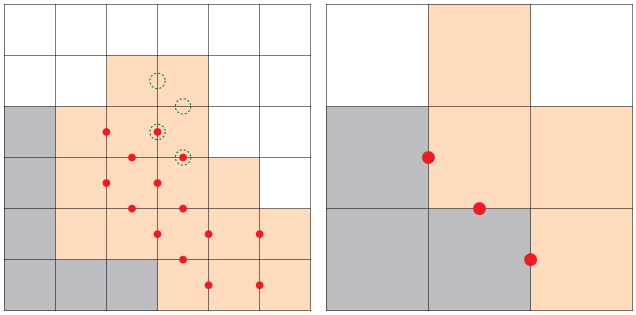

Fig. 8: *Boundary discrepancies in the fine (left) and coarse (right) domains. On the right, red dots indicate locations containing Dirichlet conditions on the coarse grid, but interior equations on the fine grid. On the left, red dots indicate interior equations that would restrict residuals on one or more locations occupied by Dirichlet conditions on the coarse grid; those restricted residuals will be replaced with zero. Green circles indicate fine interior variables that prolongate their correction from boundary coarse variables.*

ometrical change of the Dirichlet region, certain coarse Dirichlet equations will be centered on locations that were interior in the fine grid (shown as red dots in Figure 8, right). The fine grid interior equations (red dots in Figure 8, left) that would restrict their residuals onto these (now Dirichlet) coarse locations, will not have their residuals well represented on the coarse grid. We compensate for this inaccuracy by performing an extra 2-3 sweeps of our boundary Gauss-Seidel smoother over these equations, driving their residuals very close to zero, just prior to restriction. A similar inaccuracy may affect the prolongation of the correction: as we previously mentioned, the active region may have extended more in the coarse grid, compared to the fine. This discrepancy may introduce inaccuracies in the coarse grid solution near such relocated boundaries. Again, we compensate by performing an additional 2-3 Gauss-Seidel smoother sweeps on the locations of the fine grid that prolongate corrections from such relocated boundary variables (depicted as green circles in Figure 8, left). This simple treatment proved quite effective to guarantee a good coarse correction despite the small geometrical discrepancies of the two domains.

7. CO-ROTATIONAL LINEAR ELASTICITY

In the large deformation regime, and in the presence of large rotational deformations, the linear elasticity model develops artifacts such as volumetric distortions in parts of the domain with large rotations. We provide an extension to the co-rotational linear elasticity model, which has been used in slightly different forms by a number of authors in computer graphics [Müller et al. 2002; Hauth and Strasser 2004; Müller and Gross 2004], and has also used with finite elements and multigrid by [Georgii and Westermann 2006; 2008]. The co-rotational formulation extracts the rotational component of the local deformation at a specific part of the domain by computing the polar decomposition of the deformation gradient tensor $\mathbf{F} = \mathbf{RS}$ into the rotation $\mathbf{R}$ and the symmetric tensor $\mathbf{S}$. The stress is then computed as $\mathbf{P} = \mathbf{R}\mathbf{P}_L(\mathbf{S})$, where $\mathbf{P}_L$ denotes the stress of a *linear* material, as described in equation (1). Thus, the co-rotational formulation computes stresses by applying the constitutive equation of linear elasticity in a frame of reference that is rotated with the material deformation as follows:

$$\begin{aligned}\mathbf{P} &= \mathbf{R}\mathbf{P}_L(\mathbf{S}) = \mathbf{R}\left[2\mu(\mathbf{R}^T\mathbf{F}-\mathbf{I}) + \lambda\mathrm{tr}(\mathbf{R}^T\mathbf{F}-\mathbf{I})\mathbf{I}\right] \\ &= 2\mu(\mathbf{F}-\mathbf{R}) + \lambda\mathrm{tr}(\mathbf{R}^T\mathbf{F}-\mathbf{I})\mathbf{R} \\ &= 2\mu\mathbf{F} + \lambda\mathrm{tr}(\mathbf{R}^T\mathbf{F})\mathbf{R} - (2\mu+d\lambda)\mathbf{R} \\ &= 2\mu\mathbf{F} - \mu p\mathbf{R} - (2\mu+d\lambda)\mathbf{R} \end{aligned} \quad (13)$$

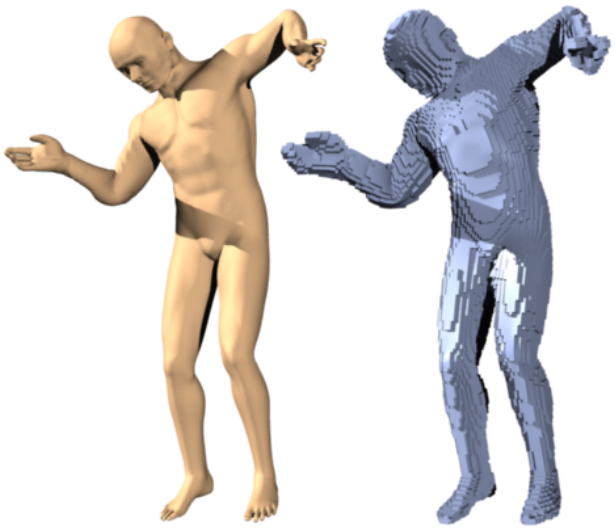

Fig. 9: *Simulation of a human character driven by a kinematic skeleton. The high-resolution rendering surface is seen in the left, while the simulation lattice is depicted on the right (resolution: 142K nodes, grid spacing 9mm).*

where the last form of the stress in equation (13) results from introducing an auxiliary pressure variable $p{=}{-}(\lambda/\mu)\mathrm{tr}(\mathbf{R}^T\mathbf{F})$ similar to the augmentation used for linear elasticity in section 4.1. As before, the augmented position equations are defined as $\partial_j P_{ij}{=}f_i$. Combining with the pressure equations and rearranging we get

$$\begin{pmatrix} 2\mu\Delta\mathbf{I} & -\mu(\boldsymbol{\nabla}^T\mathbf{R}^T)^T \\ \mu(\mathbf{R}\boldsymbol{\nabla})^T & \frac{\mu^2}{\lambda} \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ p \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}-(2\mu+d\lambda)\boldsymbol{\nabla}\cdot\mathbf{R} \\ 0 \end{pmatrix}. \quad (14)$$

The notation for the off-diagonal blocks of the matrix in equation (14) was used to indicate whether the operators $\boldsymbol{\nabla}, \boldsymbol{\nabla}^T$ operate or not on the rotation matrix $\mathbf{R}$. In index form, these operators equal $[\mu(\boldsymbol{\nabla}^T\mathbf{R}^T)^T]_i = \mu\partial_j R_{ij}$, and $[\mu(\mathbf{R}\boldsymbol{\nabla})^T]_i = \mu R_{ij}\partial_j$ respectively. In contrast with the equations of linear elasticity, equation (14) is a nonlinear PDE, since both the operator matrix and the right hand side vector contain the rotation matrix $\mathbf{R}$ which depends on the current deformation $\boldsymbol{\phi}$ itself. We highlight this fact by writing this system as $\mathcal{L}[\boldsymbol{u}]\boldsymbol{u}{=}\boldsymbol{f}[\boldsymbol{u}]$. Nevertheless, for the purposes of a multigrid scheme it is possible to treat system (14) as a linear equation, by freezing the values of $\mathcal{L}$ and $\boldsymbol{f}$ for the duration of a V-cycle, and updating them after a better solution to this frozen coefficient system has been obtained. In an iterative fashion, we obtain the $(k{+}1)$-th approximation to the solution of the linear system by executing one V-cycle on the constant coefficient system $\mathcal{L}[\boldsymbol{u}^k]\boldsymbol{u}^{k+1}{=}\boldsymbol{f}[\boldsymbol{u}^k]$ (or quasi-linear form) of equation (14) in this context. We generalize the distributive smoothing approach to the quasi-linear equation (14). In this case, the distribution matrix is

$$\mathcal{M} = \begin{pmatrix} \mathbf{I} & -(\boldsymbol{\nabla}^T\mathbf{R}^T)^T \\ \mathbf{0} & -2\Delta \end{pmatrix}. \quad (15)$$

Then, the distributed operator $\mathcal{LM}$ becomes

$$\mathcal{LM} = \begin{pmatrix} 2\mu\Delta\mathbf{I} & 2\mu\left[(\boldsymbol{\nabla}^T\mathbf{R}^T)^T\Delta - \Delta(\boldsymbol{\nabla}^T\mathbf{R}^T)^T\right] \\ \mu(\mathbf{R}\boldsymbol{\nabla})^T & -\mu(1+\frac{2\mu}{\lambda})\Delta \end{pmatrix} \quad (16)$$

The top right block of $\mathcal{LM}$ would be equal to zero if $\mathbf{R}$ is a spatially constant rotation, but not in the general case. However, near a solution where the rotations are expected to be smooth, this value is effectively zero, and $\mathcal{LM}$ becomes a triangular matrix, similar to the linear case. Effectively, even if the distributed system is near-triangular, a Gauss-Seidel algorithm will still be an acceptable smoother. In practice we found distributive Gauss-Seidel to be a good smoother for the quasi-linear problem at all times, although the convergence rates were slightly lower away from the solution.

For the purposes of boundary smoothing, we again derive a symmetric definite discretization where Gauss-Seidel can be used. The equations are written in the divergence form $\mathcal{L}_i\boldsymbol{u}=\partial_j P_{ij}$ and any exterior stresses are eliminated from the divergence stencil using an appropriate traction equation, as in section 5.3. The only complication is due to the fact that pressures are now multiplied by the non-diagonal matrix $\mathbf{R}$ in the augmented stress definition (13), thus off diagonal stresses P_{ij} $(i \neq j)$ require an edge centered pressure value. Since all exterior stresses have been removed, the four incident cells to the edge in question are interior. Thus, we compute the needed pressure value by averaging these four neighboring pressures. Finally, pressure equations are written as $\mu R_{ij}\phi_{i,j} + (\mu^2/\lambda)p = 0$, indicating that all gradient values $\phi_{i,j}$ are needed at a cell center. The stencil for off-diagonal gradients will be averaged from the four neighboring edge centers, where they are naturally defined. If any such gradient is external, we evaluate this term as $R_{ij}F_{ij}$ as a frozen coefficient and move it to the right hand side. The resulting discrete system of equations is symmetric and definite (after substitution of the pressures) and can be smoothed with a Gauss-Seidel procedure as in section 5.3.

8. DYNAMICS

The static formulation of elasticity disregards any dynamic effects. Our method, however, can easily accommodate the simulation of dynamic deformation; the effect of inertia actually improves the conditioning of the discrete equations. For example, a backward Euler implicit scheme updates positions as

$$\left(\mathbf{I} - \frac{\Delta t}{m}(\Delta t + \gamma)\mathcal{L}\right)\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \Delta t\boldsymbol{v}^n - \frac{\gamma\Delta t}{m}\mathcal{L}\boldsymbol{x}^n.$$

Here, γ is the damping coefficient. Velocities are updated as $\boldsymbol{v}^{n+1}=(\boldsymbol{x}^{n+1}-\boldsymbol{x}^n)/\Delta t$. Up to scaling, this is equivalent to solving the problem $\mathcal{L}\boldsymbol{u} - c\boldsymbol{\phi} = \boldsymbol{b}$. This system can also be augmented, by adding the term $-c\mathbf{I}$ to the position equations of (5). Distributive smoothing can be followed as before, where the bottom right term -2Δ in the distribution matrix is now replaced by $-2\Delta + c/\mu$. All other formulations hold unchanged and convergence for this system will be at least as good as the static case.

9. RESULTS AND EVALUATION

9.1 Evaluation of solver performance

We first compare the performance of our method with a Conjugate Gradients (CG) solver, as illustrated in Figure 11. The left figure plots the reduction of the residual for our synthetic test model: a rectangular elastic box under mixed boundary conditions (also depicted in figure 20, on the right). We use CG to solve the symmetric, definite system resulting from the discretization of the (non-augmented) PDE (3) using finite differences on staggered grids of sizes 32^3 and 64^3, for two different values of Poisson's ratio ν. We observe that, after some initial progress, the convergence of CG slows down significantly. This deterioration is more pronounced on cases with more degrees of freedom, and higher incompressibility (which are the focus points of our method). Replacing the finite difference discretization with trilinear, hexahedral finite elements (middle plot) still exhibits the same stagnation, particularly for the more incompressible case. Our method (right plot), exhibits a practically constant convergence rate all the way until the error is at the levels of the floating point round-off threshold.

Discretization used for Multi-Grid solver and benchmark description	Poisson's ratio	Convergence rate (V-cycle)
3D Poisson equation (scalar), 64^3 periodic domain, lexicographical Gauss-Seidel smoothing (reference problem)	N/A	0.19
Non-augmented linear elasticity, staggered finite difference discretization, 32^3 elastic box, mixed BCs	0.2	0.41
	0.475	0.85
	0.49	0.9
Linear elasticity, tetrahedral finite element discretization, 32^3 elastic box (160K tetrahedra), mixed BCs	0.2	0.35
	0.475	0.8
	0.49	0.89
Our method: Distributive Gauss-Seidel on augmented PDEs 32^3 elastic box, mixed BCs	0.2	0.26
	0.49	0.28
Our method on the 302K vertex Armadillo model (using linear elasticity)	0.475	0.21-0.35 (typical)
		0.62 (asymptotic)
Our method on the 302K vertex Armadillo model (co-rotational linear elasticity)	0.475	~0.38 (linear residual)
		~0.43 (nonlinear residual)
Our method on the 1.1M vertex articulated Human model	0.475	0.24
Our method on the 43K car model (backward euler & dynamics)	0.4	0.08

Fig. 10: *Comparison with alternative multigrid techniques. Convergence rate is defined as the asymptotic residual reduction factor:* $|r_{k+1}|/|r_k|$

We subsequently compare our method with other multigrid techniques (i.e. using different relaxation or discretization approaches), in Figure 10. As a general comment, all methods evaluated here were able to achieve convergence rates that are largely independent of the model resolution (in contrast with CG). As a point of reference we include the convergence rate of 0.19 of a periodic 3D Poisson problem with lexicographical Gauss-Seidel smoothing. We first experimented with a staggered finite difference discretization of equation (3) which did not however employ the augmentation of section 4.1. We observe that the convergence rate is deteriorating with higher incompressibility, to reach a value of 0.9 for a material with $\nu = 0.49$. A similar behavior is observed with a tetrahedral finite element discretization, used in place of finite differences. These results are compatible with the findings of [Griebel et al. 2003] who observe similar problems with near-incompressibility even for AMG solvers. Our method exhibits convergence rates of 0.26-0.28 even for highly incompressible materials. Apart from the convergence experiments performed on our synthetic elastic box example, figures 10 and 11 include experiments performed on irregular geometries such as the armadillo model of figure 24 and the human character of figure 9. We further discuss the convergence rates and run times of these irregular models in section 9.3.

9.2 Discretization accuracy analysis

Our method simulates objects of irregular shapes by embedding them in regular Cartesian lattices. Embedded simulation has been a popular method for physics-based animation, using either Cartesian lattices [Müller et al. 2004; Rivers and James 2007] for sim-

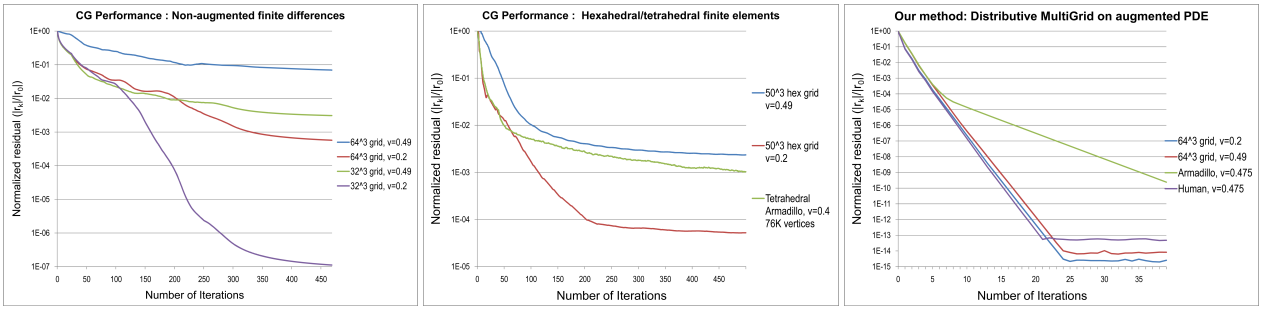

Fig. 11: *Comparison of a CG solver on finite difference (left) or finite element (middle) discretizations, with our proposed method (right)*

plicity and efficiency, or tetrahedral embedding meshes for applications such as biomechanics [Lee et al. 2009] and fracture modeling [Molino et al. 2004]. Although embedded models are computationally efficient and easy to generate, conforming meshes generally approximate the surface geometry of a model better than embedded models of comparable resolution. Several authors have proposed methods to compensate for this effect, for example by resolving collision and surface dynamics at a sub-element level [Sifakis et al. 2007] or using an alternative interpolation method to generate the embedded surface for rendering [Kaufmann et al. 2009]. In this section we evaluate the accuracy of our embedded method against a conforming discretization, and also compare our finite-difference method to an embedded discretization using finite elements.

For our accuracy analysis, we construct a 2D elasticity problem with an analytically known solution. Our testing model is the disc $D = \{(x, y) : (x - 0.5)^2 + (y - 0.5)^2 \leq 0.25^2\}$ and is deformed according to the deformation function $\boldsymbol{\phi}(\boldsymbol{X})=(\phi_1, \phi_2)$ defined as

$$(\phi_1(x, y), \phi_2(x, y)) = \frac{2x}{\sqrt{\pi}} \left(\cos\left(\frac{\pi y}{2}\right), \sin\left(\frac{\pi y}{2}\right) \right) \qquad (17)$$

We assume a linear elastic material. We substitute this analytic form of the deformation function $\boldsymbol{\phi}$ into the linear elasticity equations (3) to obtain the analytic expression of the elastic forces $\boldsymbol{f}$. We treat two quadrants of the outline of D (the thick shaded curves in figure 12) as Dirichlet boundary conditions, while the rest of the boundary (the two unshaded quadrants of the outline) are treated as traction boundaries. We analytically compute the traction value along the circular boundary of D as $\boldsymbol{t} = \mathbf{PN}$ where the stress $\mathbf{P}$ is computed from equation (1) and $\mathbf{N}$ is the outward pointing normal. Despite its geometric simplicity, this test problem highlights certain challenges related to embedding and discretization, especially for large values of Poisson's ratio, since the deformation prescribed in equation (17) incurs substantial change of volume in different parts of the domain (as seen in figure 12) giving rise to large elastic forces. We compare our embedded finite difference discretization with two finite element discretizations, one defined on a conforming tessellation of D, and one using an embedding triangular mesh, as illustrated in figure 13.

Since our test problem involves non-zero traction conditions on the embedded boundary (in contrast with our other examples in this paper which use free boundaries, with zero traction) we need to treat this traction condition properly for the embedded finite element or finite difference grids. For these embedded discretizations, we start by approximating the circular boundary of D (the green curve in figure 13) with a polygonal curve. We subsequently compute a force for each segment of this polygonal curve that falls within the part of the boundary where traction conditions are given. This force is computed from the traction value as $\boldsymbol{f} = l \cdot \boldsymbol{t}$ where l is the length of the segment. We distribute half of this force to each endpoint of the segment; the traction condition is thus converted into individual forces on the vertices of the embedded boundary. Finally, we remap these forces from the polygonal boundary curve back to the degrees of freedom of the embedding simulation mesh. For a triangular embedding, this is accomplished by simply distributing the force from a vertex of the embedded boundary to the three vertices of its containing simulation triangle, weighted by the barycentric weights of the boundary location in the triangle (see figure 14, left). In our staggered, Cartesian embedding the x and y coordinates are embedded in two non-collocated lattices; thus, we distribute the x component of the force (denoted as f_1 in 14, right) to ϕ_1 grid locations, weighted by the bilinear embedding weights of the boundary location in this grid, while the y component of the force (denoted as f_2) will be similarly distributed to ϕ_2 grid locations. After this

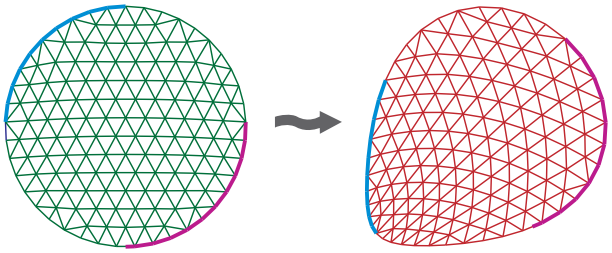

Fig. 12: *Illustration of the analytic deformation in our accuracy study. The thick shaded boundary sections indicate Dirichlet boundary conditions. The undeformed object is depicted on the left, the deformed state on the right.*

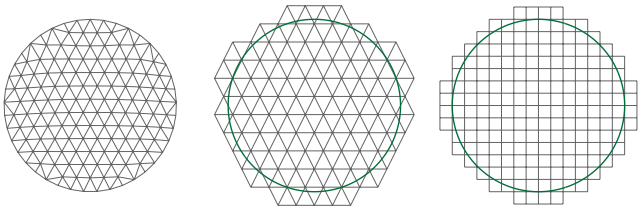

Fig. 13: *The three discretization methods in our comparative study. Left: A conforming tessellation, discretized with the finite element method. Middle: An embedded finite element discretization on a triangular mesh. Right: Our staggered finite difference method based on a Cartesian background lattice.*

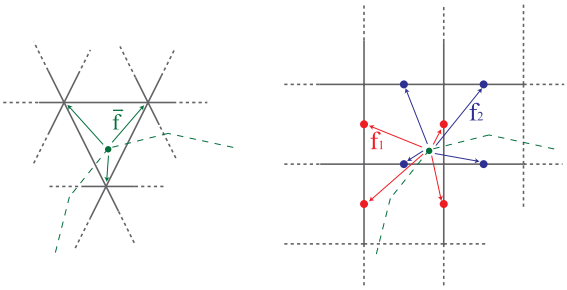

Fig. 14: *Left: Boundary traction forces are barycentrically distributed to the vertices of a triangular embedding mesh. Right: In our staggered discretization, each component of the traction force is bilinearly distributed to the grid locations of the respective (staggered) variable.*

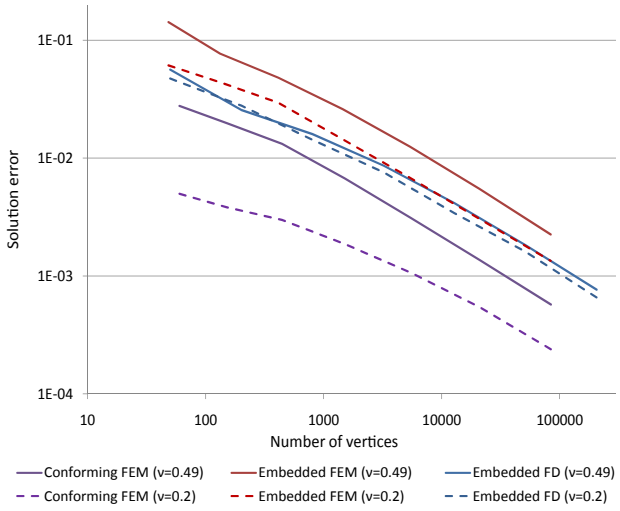

Fig. 15: *Plots of the maximum solution error for various discretizations. Solid lines indicate near-incompressible material parameters while dashed lines correspond to low Poisson's ratio. In this doubly-logarithmic plot, a slope of* -0.5 *indicates a first-order accurate method. The asymptotic order of accuracy observed from all six experiments ranges between* $1.15-1.25$.

remapping, traction forces that have been mapped to locations of interior variables are scaled by $1/h^2$ (to remove the area weighting and convert them to force densities, as in the PDE form of elasticity) and then added to the right hand side of the discrete equation $\mathcal{L}\phi = \boldsymbol{f}$, while forces mapped to boundary variable locations are converted back into traction conditions on the faces of the embedding grid as $t_i = f_i(N \cdot N')/h$, where N is the normal to the embedded boundary and N' is the normal to boundary face of the embedding grid. Notably, for free (zero-traction) boundaries, this treatment simply reduces to the method described in section 5.3.

Figure 16 illustrates the accuracy of the different discretization methods in our test, for different resolutions and degrees of incompressibility. Figure 15 plots the maximum error in the computed discrete solution under different levels of refinement. Since a discretization with order of accuracy equal to κ bounds the error as $|e| = O(h^\kappa)$ and $h \approx N^{-1/2}$ where N is the number of vertices in a uniform discretization, the asymptotic order of accuracy is approximated as $\kappa \approx -2 \log |e| / \log N = -2m$, where m is the slope of the doubly-logarithmic plot of figure 15. We emphasize that the order of accuracy assessed in this section is completely independent of the convergence rate of the solver used for each discretization (see section 9.1). The discrete problems formulated in this section were solved to full convergence with an appropriate solver (multigrid or conjugate gradients). Our findings are summarized as follows:

—Although the different discretizations under consideration start with different levels of error for a base resolution, this error is asymptotically reduced at comparable rates under refinement. We estimated an asymptotic order of accuracy between $1.15-1.25$ from the tests plotted in figure 15. This approximate first-order accuracy is to be expected both from our finite difference scheme (due to the first-order treatment of the boundary) and the finite element discretizations (due to the use of first-order linear triangle elements, see e.g. [Hughes 1987]).

—The conforming discretizations produced lower errors than both the finite difference, and finite element based embedded discretizations. For materials with low Poisson's ratio, our proposed embedded method would require approximately $10-20$ times more degrees of freedom to match the accuracy of the conforming discretization. For near-incompressible materials this discrepancy is smaller, with our embedded method requiring approximately one extra grid refinement to reach the accuracy of the conforming method. Of course, a comparison of the degrees of freedom necessary for a given measure of accuracy does not necessarily reflect the computational cost of each approach. Our method typically leads to significantly reduced run times compared to conforming tetrahedral FEM models with the same number of degrees of freedom, due to the regularity of the discretization, convergence efficiency of the multigrid solver and improved numerical conditioning from our treatment of near-incompressibility. These performance benefits are less evident for low-resolution models (with up to a few thousand of degrees of freedom) where a conforming model, if available, may offer better accuracy per computation cost. For large models such as the ones demonstrated in our examples in section 9.3 our method can significantly outperform conforming tetrahedral meshes for the same degree of accuracy, even if our method requires a higher number of degrees of freedom to achieve the same accuracy. Finally, generating a good conforming tetrahedral model for detailed geometries such as those in figures 25 and 27 is a nontrivial meshing task which is not necessary in our embedded scheme.

—We also observe a tendency for the error on the embedded discretizations to be more oscillatory near the boundary, compared to the conforming case. These embedding artifacts are typically less pronounced with our method than with an embedded finite element approach on near-incompressible materials (see figure 16), they are attenuated under refinement and can be significantly reduced in practice by slightly padding the embedding mesh outwards, typically by as little as one layer of cells (see figure 17).

—Our method matches or exceeds the accuracy of the embedded finite element discretization in our tests. The two embedded methods yield comparable accuracy for materials with low Poisson's ratio, especially in the asymptotic limit. For modest to high degrees of incompressibility, our method is noticeably more accurate and less prone to embedding artifacts than the embedded finite element discretization at the same resolution. Finally, our method performs similarly for materials of low and high incompressibility, although the embedded boundary surface tends to be slightly smoother for compressible materials.

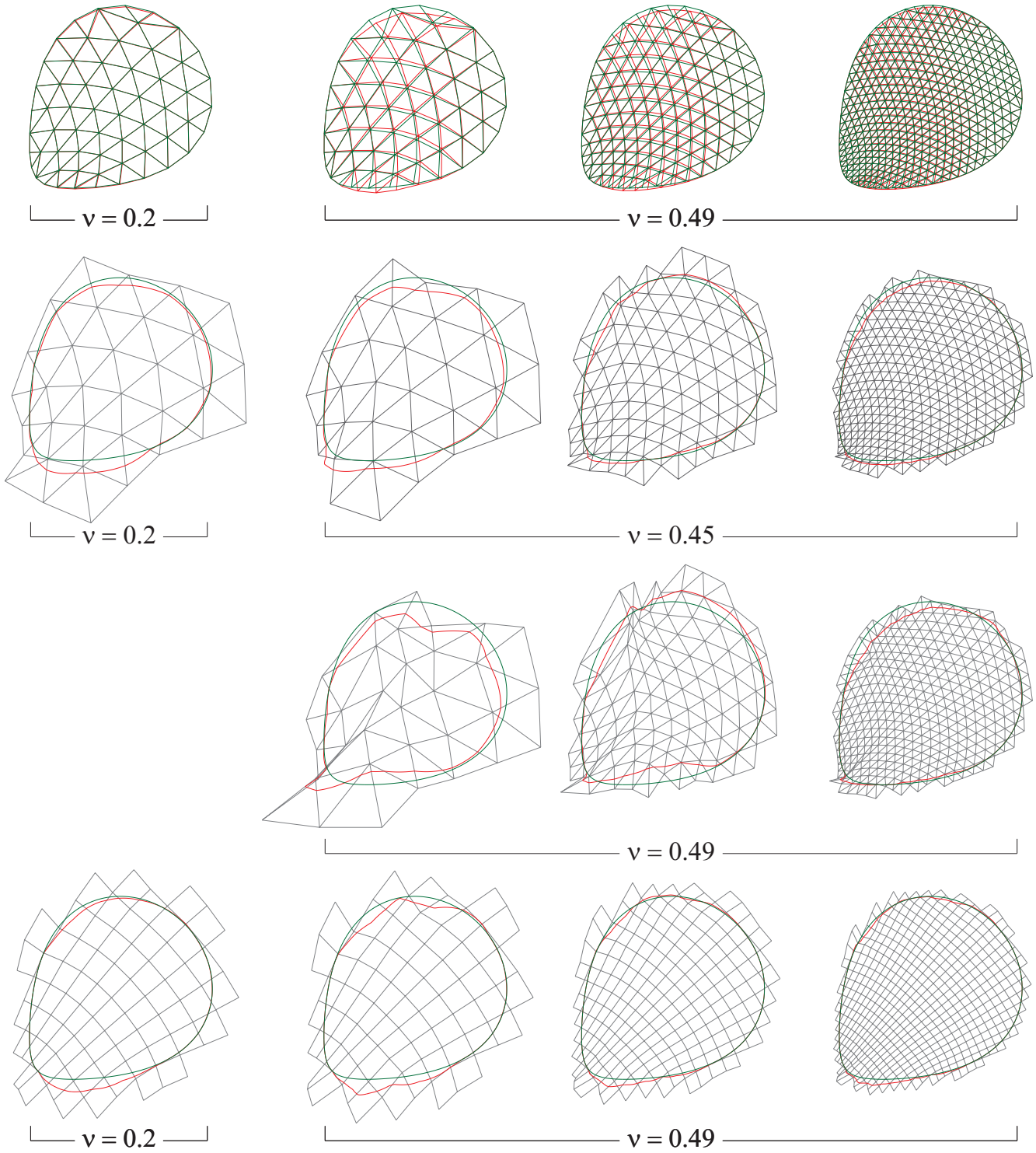

Fig. 16: *Convergence of different discretizations under refinement. The analytic solution is depicted in green, the discrete solution in red. The Poisson's ratio (ν) used for each experiment is given. Top row: A finite element discretization on a conforming triangle mesh. Compressible (ν=0.2) and near-incompressible (ν=0.49) cases are shown. Second and third row: Embedded finite element simulation on a triangle mesh. An additional case of moderate incompressibility (ν=0.45) is illustrated. Bottom row: Our embedded finite difference method. Note that both the embedded boundary and the background lattice are independently interpolated from the staggered deformation variables (not pictured). Also, the resolution in the rightmost column corresponds to approximately the same number of degrees of freedom for all discretizations.*

Two important caveats should also be mentioned: The circular elastic body in our test had a smooth boundary surface which was well approximated by conforming tesselations even at low resolutions. Highly detailed models with intricate features (see e.g., figures 25 and 27) would incur significantly higher approximation errors for a conforming tesselation that does not descend to the resolution level necessary to resolve all the geometric detail. Secondly, in our embedded examples we used the analytic expression of the deformation field in equation (17) to specify Dirichlet boundary conditions directly on the vertices of the embedding meshes. This can be an acceptable practice for applications such as skeleton-driven characters, where kinematic constraints have a volumetric extent and can therefore be sampled at the locations of the simulation degrees of freedom. However, when Dirichlet conditions need to be specified at sub-grid locations and extrapolating these constraints to simulation vertices is not convenient, conforming meshes that resolve the constraint surface would be at an advantage. In future work we will investigate adding embedded soft-constraints in our framework (see e.g. [Sifakis et al. 2007]) to provide this additional flexibility. Finally, in our tests we considered discretizations of approximately uniform density (even when the mesh topology was irregular). It is also possible to use an adaptive discretization, either in the form of an adaptive conforming tesselation or an adaptive finite difference scheme (see e.g. [Losasso et al. 2004]). In fact, there are well established multigrid methods that operate in conjunction with adaptive discretizations [Brandt 1977], and we believe the elasticity solver proposed in this paper can be similarly applied to adaptive (e.g. octree) discretizations. We defer this extension to future work, along with a principled comparative evaluation of different adaptive discretization schemes for elasticity, especially in light of the nontrivial implications adaptivity may have on accuracy, numerical conditioning and potential for parallelization.

9.3 Animation tests

In addition to our comparative benchmarks, we tested our method on models with elaborate, irregular geometries. Figure 9 demonstrates the simulation of flesh of a human character with keyframed skeleton motion. The model was simulated at 2 resolutions yielding V-cycle times of 0.62sec for a 142K vertex model (pictured in figure 9), and 3.48sec for a larger resolution with 1.15M vertices (figure 17, right). The convergence rate for this example, as seen in Figures 11(right) and 10, was slightly better than our synthetic box examples at 0.24. We attribute this result to the extensive Dirichlet regions throughout the body induced by the kinematic skeleton, which stabilize the model and allow for highly efficient smoothing. In contrast, the armadillo model of figure 24 is very weakly constrained, with Dirichlet regions defined only over the hands and feet (see also [Griebel et al. 2003] for a discussion of sub-optimal smoothing performance with dominating traction boundaries). In this model with extensive zero-traction boundary conditions, our method exhibited convergence rates between 0.21-0.35 for the first 7-8 V-cycles after a large perturbation; at that point the residual had been reduced by four orders of magnitude and the model had visually reached convergence after just the first few iterations. Subsequent V-cycles would ultimately settle at an asymptotic rate of 0.62 which could be improved by increasing the intensity of the boundary smoother, although this was not pursued since the model was already well converged and the extra smoothing cost would not be practically justified. With typical incremental motion of the boundary conditions, 1-2 V-cycles per frame would be enough to produce a visually converged animation.

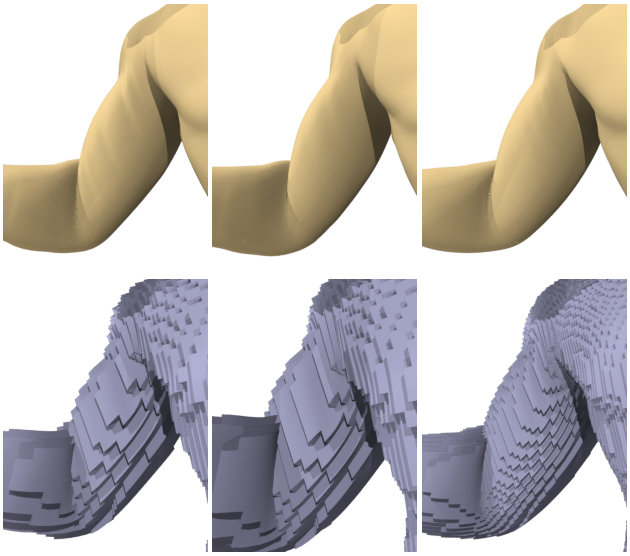

Fig. 17: *Closeup of the elbow joint from figure 9. Left: Grid spacing 9mm (142K vertices). Embedding artifacts are visible on the surface. Middle: Padding the embedding cage with one additional layer of cells visibly reduces the artifacts. Right: Surface artifacts are outright reduced using a higher resolution embedding cage (4.5mm spacing, 1.15M vertices).*

Figure 10 also reports the convergence rates for the armadillo model of figure 24 simulated using co-rotational linear elasticity. Since the coefficients of the discrete system vary with the current configuration, the convergence rate is also variable. Additionally, the residual of the quasi-linearized system will differ from its actual non-linear counterpart; this discrepancy will also depend on whether the quasi-linearization process is close to convergence. The rates reported are typical of the animations shown, assuming 2 V-cycles per frame, and update of the quasi-linearization every 5 V-cycles. The average run time was 5.1sec per V-cycle, 10.2sec/frame (with 2 V-cycles). For comparison, we also simulated the tetrahedral armadillo model of [Teran et al. 2005] using the quasistatic solver described in their paper. This tetrahedral model contains 380K tets and 76K vertices, thus contains approximately one quarter of the degrees of freedom of our embedded model in figure 24. For a Poisson's ratio of 0.4 each Newton-Raphson iteration (which includes a CG solve) required approximately 8.7sec while 5 Newton iterations per frame were required for acceptable convergence, leading to an approximate cost of 43.5sec/frame.

We also demonstrate examples of fully dynamic simulation. In figure 26, a 43K vertex car model is simulated using the static elasticity equations, as well as the dynamic scheme of section 8. As expected, the convergence rate for the Backward Euler system was significantly faster than our static problem (due to the addition of the identity term in the system matrix). Using a time step Δt equal to the frame time, our observed convergence rate was 0.08. Figure 27 illustrates the dynamic simulation of an elastic dragon figurine. The embedding grid has 402K cells/voxels and simulation cost is 8.2sec/frame. Figure 25 illustrates a high-velocity impact of a rigid body on a face model. The embedding grid contains 915K cells/voxels and simulation cost is 21sec/frame. We note that no explicit collision handling was performed for this example; instead, the degrees of freedom of the face that came in contact with the impacting object were kinematically prescribed to move with it for the

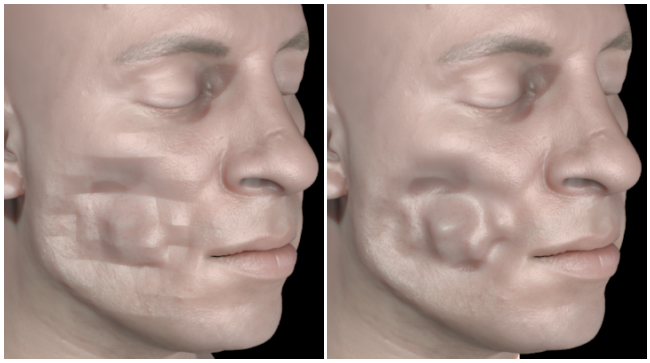

Fig. 18: *Comparison of trilinear (left) and tricubic interpolation (right) on a coarse simulation. The embedding grid includes only 11K cells.*

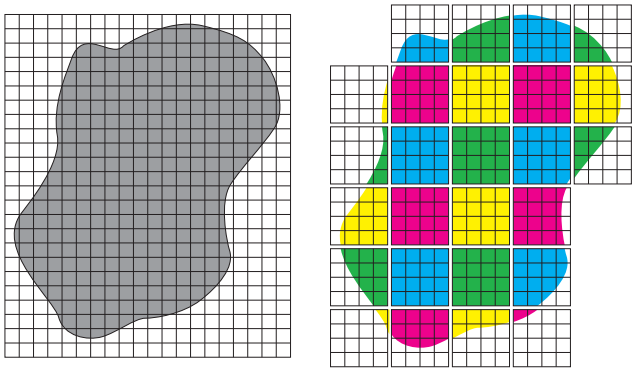

Fig. 19: *Volumetric partitioning using colored blocks in a 2D domain.*

duration of the impact. For these dynamic simulations just a single V-cycle per frame was sufficient, due to the better conditioning of the Backward Euler equations. Additionally, figure 22 provides a detailed breakdown of the execution cost of the individual subroutines on some of our benchmarks.

Finally, we note that usual trilinear interpolation would infrequently give rise to visual artifacts. Such artifacts would surface in simulations where the resolution of the embedding grid was substantially coarser than that of the embedded surface (see, e.g. figure 18) and in conjunction with very extreme deformation. Since trilinear interpolation does not produce continuous derivatives, surface normals would exhibit visible discontinuities in these cases. We found that using tricubic interpolation as in [Lekien and Marsden 2005] effectively eliminates this problem, as seen in figure 18. Notably, their method is based on evaluating higher order derivatives at the nodes of the interpolation lattice, a process that is trivially implemented with finite differences in our regular discretization.

9.4 Parallelization

Our discretization of elasticity and the multigrid solver proposed in our paper possess a number of characteristics that favor parallelism and scalable performance. The use of regular grids promotes locality, allows operations such as smoothing and transfer between grids to be implemented as streaming operations and allows for easy domain partitioning based on the background grid. Indirect memory access is avoided since we do not use an explicit mesh to represent the simulated model. In addition, the regularity of the discrete equations enables a compact storage of the matrix in our linear system. For example, in the case of linear elasticity, all interior equations use the same stencil, thus there is no need to store a separate equation per grid location; this allows for a small memory footprint, even for large domains (see figure 23). In the case of co-rotational linear elasticity, only the rotation field needs to be stored and updated on the grid, while the system matrix can be built on-the-fly. We evaluated the potential of our algorithm for parallel performance by multi-threading a specific test problem on shared memory platforms. In this section we describe our parallelization methodology and present performance measurements.

Parallelization of the components of the multigrid solver is based on an appropriate domain decomposition. Operations such as the interior (distributive) smoother, the computation and restriction of residuals, and prolongation of the coarse-grid correction are performed throughout the volumetric extent of the simulated model. Consequently, these subroutines require a volumetric partitioning of the simulation domain. This is simply accomplished by partitioning the background cartesian grid into rectangular blocks, as depicted in the two-dimensional illustration of figure 19, which can then be processed by separate threads. However, operations such as the smoother incur data dependencies between neighboring blocks. In lieu of locking, we employ a coloring of the blocks (4 colors are used in the two-dimensional example of figure 19, 8 colors would be used in 3D) such that no two blocks of the same color are neighboring. All blocks of the same color can be processed in parallel without data dependencies, while different color groups are processed in sequence. The optimal block size depends on the desired number of blocks per color group (to allow simultaneous use of more processing threads) and the properties of the memory subsystem. For example, rectangular blocks of $16 \times 8 \times 8$ cells will align well with 512-bit cache lines ($16 \times$ 4-byte floats) while providing a few tens of blocks per color for problems in the order of 10^5 nodes.

Additionally, the rectangular shape of these blocks simplifies their traversal, as this can simply be performed with a triple loop over fixed index ranges. Such static loops yield improved cache and prefetching performance, and facilitate vectorization (either explicitly or as a compiler optimization). Care needs to be taken for blocks that include the domain boundaries, since some of their cells are outside the active simulation domain. In order to retain the benefits of traversing the block with a static triple loop, we perform the operation in question (i.e. smoothing, residual computation, intergrid transfer) for all cells of a block, but only write the output of this operation conditionally on the value of a bitmap that indicates active/inactive grid locations. Finally, there are certain subroutines of our multigrid solver (e.g. boundary smoothing) that operate on a band around the surface of the object. Since a volumetric partitioning could be inefficient and unbalanced for these operations, we perform a separate partitioning of the surface of the object. A chromatic grouping of these surface partitions is precomputed, as seen in figure 20, to allow all blocks withing a color group to be processed in parallel without locking.

We evaluated the potential of our algorithm for parallel performance using a multi-threaded version of our solver on the following shared-memory platforms: An 8-core SMP workstation with 3.0GHz Intel X5365 Xeon processors, a 16-core SMP server with 2.93GHz Intel X7350 CPUs, and a cycle-accurate performance simulator for the upcoming x86-based many-core Intel architecture codenamed Larrabee. Our benchmarks were based on our synthetic

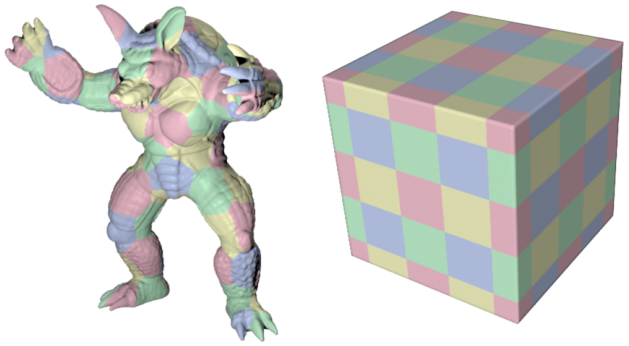

Fig. 20: *Surface partitioning of 3D models into colored surface patches.*

elastic box example, under high incompressibility ($\nu = .48$), with mixed boundary conditions and at resolutions ranging from 32^3 to 256^3 vertices. Figure 23 illustrates the speedup of our benchmarks, and associated working set sizes, on the 8- and 16-core SMPs. Figure 21 illustrates the speedup of individual subroutines on the Larrabee simulator, for two different problem sizes and at configurations up to 32 cores, with 4 threads/core. We note that the utilization of more than 1 thread per core does not increase the computational bandwidth (instructions are sequentially dispatched from different threads), but serves to hide instruction and memory latencies. Although we did not exploit the SIMD capacity of Larrabee in this experiment, the memory utilization was at a low 0.5GB/Gcycles per core, demonstrating there is a substantial memory bandwidth margin to allow vectorization to further improve the performance of our solver.

10. CONCLUSION

Our multigrid framework allows for the efficient simulation of deformable materials with many degrees of freedom over a wide range of material parameters including the near-incompresible limit and our finite difference discretization naturally accommodates arbitrary irregular geometries. However, the strength of the approach would be improved with the generalization to arbitrary hyperelastic constitutive models. We also plan to investigate efficient self-collision detection and handling techniques that will not become a performance bottleneck, given the efficiency of the elasticity solver. Multi-resolution collision detection and response techniques (e.g. [Otaduy et al. 2007]) would be expected to be the most compatible candidates. In addition, if a penalty-based collision response were desired, the embedded treatment of non-zero traction conditions described in section 9.2 can be used to apply those penalty forces at sub-cell resolution. Finally, although our initial investigation has demonstrated excellent potential for scaling on many-core platforms, a more principled investigation needs to assess the performance of our method platforms with SIMD capability, and address a broader spectrum of constitutive behaviors and interacting geometries.

CMP Scalability -- Problem size 64x64x64										
Kernel	Core 2	1c1t	1c4t	4c1t	4c4t	8c1t	8c4t	16c1t	32c1t	32c4t
Boundary smoothing	2.98	1.00	1.87	3.76	6.52	7.00	12.07	20.84	22.67	33.43
Distributive smoothing	2.96	1.00	1.84	3.90	6.86	7.55	13.44	13.79	28.37	24.51
Residual computation	3.79	1.00	1.85	3.95	7.24	7.82	14.11	14.75	29.61	26.58
Restriction	2.66	1.00	1.74	3.76	6.08	6.81	9.82	15.80	17.64	25.84
Prolongation	3.51	1.00	1.77	3.87	3.78	4.33	3.74	5.64	8.27	7.16
CMP Scalability -- Problem size 32x32x32										
Kernel	Core 2	1c1t	1c2t	1c4t	4c1t	4c4t	8c1t	16c1t		
Boundary smoothing	2.06	1.00	1.59	1.70	3.23	5.74	5.61	10.52		
Distributive smoothing	1.75	1.00	1.65	1.80	3.72	6.20	6.79	11.59		
Residual computation	2.42	1.00	1.63	1.76	3.66	6.31	7.08	12.55		

Fig. 21: *Parallel scaling on a Larrabee simulator for a number of different configurations. $NcMt$ indicates a simulated platform with N cores and M threads/core. Prolongation/restriction were serialized on grids 32^3 and smaller. "Core 2" indicates the speedup of a single-threaded execution on an Intel Core 2 processor at the same clock frequency as the simulated platform.*

Single-core execution times of individual subroutines (in seconds)	32^3 Elastic Box (5 levels)	64^3 Elastic Box (6 levels)	128^3 Elastic Box (7 levels)	38K vertex Armadillo (4 levels)	302K vertex Armadillo (5 levels)	302K vertex Armadillo (corotational elasticity)	142K vertex Human (5 levels)	1.1M vertex Human (6 levels)
Boundary smoothing	0.052	0.197	0.711	0.217	0.890	1.281	0.275	1.074
Distributive (interior) smoothing	0.034	0.235	1.328	0.078	0.562	1.016	0.200	1.434
Residual computation	0.015	0.097	0.686	0.023	0.170	0.284	0.077	0.575
Residual restriction	0.001	0.012	0.088	0.002	0.015	0.014	0.010	0.079
Correction prolongation	0.007	0.040	0.240	0.011	0.096	0.099	0.036	0.294
Exact solver (at coarsest grid)	0.004	0.005	0.006	0.004	0.004	0.004	0.006	0.013
Approximate solution restriction	N/A	N/A	N/A	N/A	N/A	0.015	N/A	N/A
Update of nonlinear operator	N/A	N/A	N/A	N/A	N/A	2.37(*)	N/A	N/A
Other	0.001	0.001	0.002	0.001	0.001	0.006	0.012	0.014
Total	0.114	0.588	3.062	0.336	1.737	5.089	0.616	3.483

Fig. 22: *Single-core execution profiles. (*) The cost of the operator update was amortized based on 1 update every 5 V-cycles*

REFERENCES

BARBIC, J. AND JAMES, D. 2005. Real-time subspace integration of st. venant-kirchoff deformable models. *ACM Trans. Graph. (SIGGRAPH Proc.) 24,* 3, 982–990.

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRODER, P. 2003. Sparse matrix solvers on the gpu: Conjugate gradients and multigrid. *ACM Trans. Graph. (SIGGRAPH Proc.) 22,* 3, 917–924.

BONET, J. AND WOOD, R. 1997. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

BRANDT, A. 1977. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation 31,* 138, 333–390.

BRANDT, A. 1986. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation 19,* 1-4, 23–56.

BRANDT, A. 1994. Rigorous Quantitative Analysis of Multigrid, I: Constant Coefficients Two-Level Cycle with L_2-Norm. *SIAM Journal on Numerical Analysis 31*, 1695–1695.

Grid Size	32^3		64^3		128^3		256^3	
Threads	X5365	X7350	X5365	X7350	X5365	X7350	X5365	X7350
1	0.114	0.1172	0.588	0.624	3.06	3.12	18.27	18.61
2	0.0986	0.101	0.447	0.471	1.99	2.09	10.8	11.62
4	0.0793	0.0919	0.25	0.466	1.13	1.36	6.22	6.71
8	0.0709	0.0807	0.164	0.273	0.807	0.822	4.03	4.01
16		0.083		0.198		0.596		2.75
Working Set	26MB		45MB		180MB		1.2GB	

Fig. 23: *Scaling performance of the linear elasticity multigrid solver on multiprocessor systems (Intel X5365: 3.0Ghz, 8-core, 32GB RAM, Intel X7350: 3.0Ghz, 16-core, 32GB RAM). Measurements correspond to complete V-cycle times in seconds*

BRANDT, A. AND DINAR, N. 1978. Multigrid solutions to elliptic flow problems. *Numerical methods for partial differential equations*, 53–147.

BREZZI, F. AND FORTIN, M. 1991. *Mixed and hybrid finite element methods*. Springer: Berlin.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.* ACM Press, 41–48.

DEBUNNE, G., DESBRUN, M., CANI, M., AND BARR, A. 2001. Dynamic real-time deformations using space and time adaptive sampling. In *Proc. SIGGRAPH 2001*. Vol. 20. 31–36.

DICK, C., GEORGII, J., BURGKART, R., AND WESTERMANN, R. 2008. Computational steering for patient-specific implant planning in orthopedics. In *Proceedings of Visual Computing for Biomedicine 2008*. 83–92.

GASPAR, F., GRACIA, J., LISBONA, F., AND OOSTERLEE, C. 2008. Distributive smoothers in multigrid for problems with dominating grad-div operators. *Numerical Linear Algebra with Applications 15,* 8, 661–683.

GEORGII, J. AND WESTERMANN, R. 2006. A multigrid framework for real-time simulation of deformable bodies. *Computers and Graphics 30,* 3, 408–415.

GEORGII, J. AND WESTERMANN, R. 2008. Corotated finite elements made fast and stable. In *Proceedings of the 5th Workshop On Virtual Reality Interaction and Physical Simulation*.

GOODNIGHT, N., WOOLLEY, C., LEWIN, G., LUEBKE, D., AND HUMPHREYS, G. 2003. A multigrid solver for boundary value problems using programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/Eurographics Conf. on Graphics Hardware*. 102–111.

GREEN, S., TURKIYYAH, G., AND STORTI, D. 2002. Subdivision-based multilevel methods for large scale engineering simulation of thin shells. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*. ACM New York, NY, USA, 265–272.

GRIEBEL, M., OELTZ, D., AND SCHWEITZER, M. A. 2003. An algebraic multigrid method for linear elasticity. *SIAM J. Sci. Comput 25*, 407.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 281–290.

HARLOW, F. AND WELCH, J. 1965. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Phys. Fluids 8*, 2182–2189.

HAUTH, M. AND STRASSER, W. 2004. Corotational Simulation of Deformable Solids. In *In Proc. of WSCG*. 137–145.

HUGHES, C., GRZESZCZUK, R., SIFAKIS, E., KIM, D., KUMAR, S., SELLE, A., CHHUGANI, J., HOLLIMAN, M., AND CHEN, Y.-K. 2007. Physical simulation for animation and visual effects: Parallelization and characterization for chip multiprocessors. In *Intl. Symp. on Comput. Architecture*.

HUGHES, T. 1987. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Prentice Hall.

IRVING, G., SCHROEDER, C., AND FEDKIW, R. 2007. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph. (SIGGRAPH Proc.) 26,* 3.

JAMES, D. AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph. (SIGGRAPH Proc.) 22*, 879–887.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2008. Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation*.

KAUFMANN, P., MARTIN, S., BOTSCH, M., AND GROSS, M. 2009. Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graphical Models 71,* 4, 153–167.

KAZHDAN, M. AND HOPPE, H. 2008. Streaming Multigrid for Gradient-Domain Operations on Large Images. *ACM Trans. Graph. (SIGGRAPH Proc.) 27,* 3.

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph. 28,* 4, 1–17.

LEKIEN, F. AND MARSDEN, J. 2005. Tricubic interpolation in three dimensions. *Int. J. Numer. Methods Eng 63,* 3, 455–471.

LOSASSO, F., GIBOU, F., AND FEDKIW, R. 2004. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 457–462.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.) 23*, 385–392.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.* 49–54.

MÜLLER, M. AND GROSS, M. 2004. Interactive virtual materials. In *Graph. Interface*. 239–246.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. (SIGGRAPH Proc.) 24,* 3, 471–478.

MÜLLER, M., TESCHNER, M., AND GROSS, M. 2004. Physically-based simulation of objects represented by surface meshes. In *Proc. Comput. Graph. Int.* 156–165.

NI, X., GARLAND, M., AND HART, J. 2004. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics (TOG) 23,* 3, 613–622.

O'BRIEN, J. AND HODGINS, J. 1999. Graphical modeling and animation of brittle fracture. In *Proc. of SIGGRAPH 1999*. 137–146.

OTADUY, M. A., GERMANN, D., REDON, S., AND GROSS, M. 2007. Adaptive deformations with fast tight bounds. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 181–190.

RIVERS, A. AND JAMES, D. 2007. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. (SIGGRAPH Proc.) 26,* 3.

SHI, L., YU, Y., BELL, N., AND FENG, W. 2006. A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 1108–1117.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* 81–90.

TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. 2005. Robust quasistatic finite elements and flesh simulation. *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 181–190.

TERZOPOULOS, D. AND FLEISCHER, K. 1988. Deformable models. *The Visual Computer 4,* 6, 306–331.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Computer Graphics (Proc. SIGGRAPH 87) 21,* 4, 205–214.

THOMASZEWSKI, B., PABST, S., AND BLOCHINGER, W. 2007. Exploiting parallelism in physically-based simulations on multi-core processor architectures. *EG Symposium on Parallel Graphics and Visualization*.

TROTTENBERG, U., OOSTERLEE, C., AND SCHULLER, A. 2001. *Multigrid*. San Diego: Academic Press.

WU, X. AND TENDICK, F. 2004. Multigrid integration for interactive deformable body simulation. In *International Symposium on Medical Simulation (2004). c Association for Computing Machinery, Inc*. 92–104.

Fig. 24: *Quasistatic simulation of armadillo model with co-rotational linear elasticity. Resolution: 302K cells.*

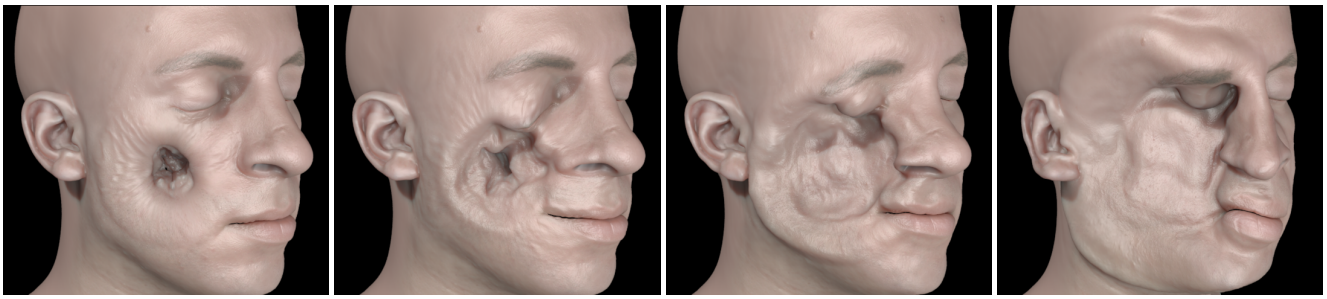

Fig. 25: *Dynamic simulation of an object impacting a face at high velocity. Modeled as a thick layer of flesh (no skull) using co-rotational linear elasticity. The embedding simulation cage contains 915K cells. The high-resolution embedded surface contains 10.1M triangles.*

Fig. 26: *Dynamic simulation of a soft elastic car model deforming under kinematic constraints, using linear elasticity. Resolution: 43K cells.*

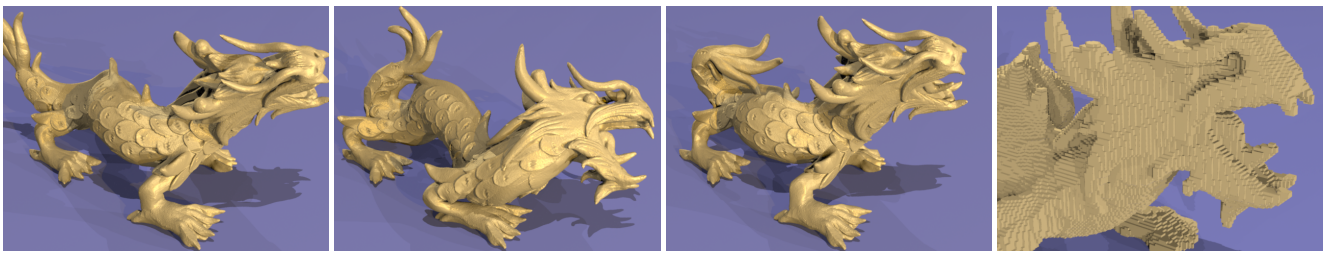

Fig. 27: *Embedded animation of a deformable dragon shaking his head, using co-rotational linear elasticity and simulation of dynamics. The embedded surface contains 7.2M triangles, while the simulated embedding cage has 402K cells (closeup pictured on the right).*