An Efficient Large Deformation Method using Domain Decomposition

Jin Huang[a*], Xinguo Liu[a†], Hujun Bao[a‡], Baining Guo[b] and Heung-Yeung Shum[b]

[a]State Key Lab of CAD&CG, Zhejiang University,
Hangzhou 310027, P.R. China

[b]Microsoft Research Asia, Sigma Center 5F, Zhichun Road No. 49, Haidian Disctrict,
Beijing, 100080, P.R. China

Efficiently simulating large deformations of flexible objects is a challenging problem in computer graphics. In this paper, we present a physically based approach to this problem, using the linear elasticity model and a finite elements method. To handle large deformations in the linear elasticity model, we exploit the domain decomposition method, based on the observation that each sub-domain undergoes a relatively small local deformation, involving a global rigid transformation. In order to efficiently solve the deformation at each simulation time step, we pre-compute the object responses in terms of displacement accelerations to the forces acting on each node, yielding a force-displacement matrix. However, the force-displacement matrix could be too large to handle for densely tessellated objects. To address this problem, we present two methods. The first method exploits spatial coherence to compress the force-displacement matrix using the clustered principal component analysis method; and the second method pre-computes only the force-displacement vectors for the boundary vertices of the sub-domains and resorts to the Cholesky factorization to solve the acceleration for the internal vertices of the sub-domains. Finally, we present some experimental results to show the large deformation effects and fast performance on complex large scale objects under interactive user manipulations.

1. Introduction

Deforming and animating soft objects have a wide range of applications in geometric modeling, computer animation, video games and surgery simulation, since many real world objects are soft and deformable. Physically based modeling has become an important approach to graphics modeling and animation. One of the simplest physically based models is the mass-spring system [1,2], which has been successfully used to simulate clothes [3–5]. Recent research trends have moved away from the simple mass-spring systems toward the more sophisticated finite element method (FEM) [6], since it is physically more accurate. Using FEM, an object's deformation behavior can be easily specified by a few material properties that have physical meanings.

There are two models in FEM based deformation approaches to measure the strain with respect to deformation in terms of displacement. One is the linear elasticity model that approximates the elastic forces as the product of a constant stiffness matrix and the displacement vector, yielding a numerically fast and stable simulation system. However, it can only model small deformations accurately. The other is the nonlinear elasticity model that models large deformations accurately with the cost of reevaluating the stiffness matrix at every time step, yielding a slow simulation and introducing numerical instabilities. By partitioning complex non-rigid behavior into global rotational motion and local deformations, a fast and stable simulation for large deformations can be obtained using the linear elasticity model [7–10].

In finite element methods, the simulated objects are tessellated into a set of elements, to approximate the continuous dynamic equations. It is easy to get hundreds and thousands of elements for objects of modest scale in graphics applications. Therefore, it is in general computationally expensive to solve the deformation at each simulation time step, and this cannot meet the interactive response requirements of graphics applications.

Our goal in this paper is to develop an efficient physically based simulation approach for large deformation of flexible objects. A preliminary version of

*Eamil: hj@cad.zju.edu.cn
†Eamil: xgliu@cad.zju.edu.cn
‡Eamil: bao@cad.zju.edu.cn

this method appeared in [11]. The basic idea behind our approach is to decompose the simulated objects into several sub-domains, such that the deformation of each sub-domain undergoes a relatively small local deformation, involving a global rigid transformation. We exploit the domain decomposition method to handle the physical interaction between the connected sub-domains. In our approach, we tessellate the volume of the simulated object as a tetrahedral mesh, and take a finite element method to solve the physical motion equation. Since the local deformation of each sub-domain is relatively small, it can be efficiently modeled with the linear elastic model.

As shown in Section 3, we need to solve a linear system for the deformation at each simulation time step. One of the most challenging issues in the domain decomposition method is how to efficiently maintain the consistency of the boundary points shared by the neighboring parts. We propose to precompute the inverse matrix of the linear system, called a force-displacement matrix, for each sub-domain, such that the next frame deformation could be obtained by matrix and vector multiplications. Since the force-displacement matrix could be too huge to handle for densely sampled objects, we further propose two methods to reduce its storage and the computation in matrix-vector multiplications. The first method compresses the force-displacement matrix using the clustered principal component analysis method [12] by exploiting its spatial coherence. The second method only records a small part of the force-displacement matrix corresponding to the boundary vertices of sub-domains, and resorts to the Cholesky factorization to solve for the internal vertices. Since Cholesky factorization maintains the matrix's sparse structure, the second method is often more efficient. An additional advantage of using precomputed force-displacement matrix is that it allows for dynamically introducing constraints for interactive user manipulation.

The previous warped stiffness methods [7,13,14] also factorize the global deformation into a global rotational deformation and a local deformation. But, they need to estimate the local rotation for each node/cell, which requires significant computational cost during simulation. Our method based on domain decomposition need only estimate the local rotation for each sub-domain, whose cost is far less than that of the warped stiffness methods. And, the stiffness matrix of each sub-domain remains constant for a rotation, which enables us to greatly improve the performance via pre-computing the force-displacement matrix. The previous modal analysis methods and the subspace integration method can achieve very high performance, but they need to reduce the DOFs of the deformation dramatically down to a very small number, which suffers a major limitation that the user constraints, e.g. position constraints, are not guaranteed to be satisfied. Our method using CPCA in Section 5.1 bears some similarity to modal analysis based methods [15] when there is only one cluster. But, we can achieve more accuracy with the same number of modes by dividing a dynamic deformation into several sub-domains.

In the rest of this paper, we will first review some related work on physically based deformation in Section 2, followed by an overview of physically based deformation with the finite element method in Section 3. Then we will introduce our domain decomposition based large deformation method in Section 4. Then we will introduce a CPCA based clustering method and a hybrid method for efficiently solving the deformation in Section 5. Finally, we will present some experimental results in Section 6 and conclude this paper with some discussion in Section 7.

2. Related Work

In the last two decades, many ingenious physically based techniques for modeling deformable objects have been proposed, which can be found in geometrical modeling, surgical simulation and computer animations. We will only review some work related to the finite element method and the linear elasticity model for solid shapes. See [16,17] for a general survey.

Finite element solvers are computationally expensive. There are many approaches for quickly solving the discretized equations using adaptive methods to avoid wasting time on minor details. Some recently proposed representative work are [18–21]. Capell et al. proposed a multiresolution framework for dynamic simulation using volumetric subdivision [18]. By adaptively refining the basis functions, Grinspun et al. proposed another simple framework for adaptive simulation [19], called CHARMS. Wu et al. [20] and

Debunne et al. [21] developed other kinds of adaptive methods using progressive meshes and LOD tetrahedral meshes. Debunne et al. also took adaptive time steps during simulation [21] to further avoid unnecessary computation.

Modal analysis with finite element methods, which decompose non-rigid dynamics into a sum of independent vibration modes, has been a well established mathematical tool in mechanical engineering [6]. By discarding the small-amplitude, high-frequency modes, an efficient and stable simulation with visually acceptable accuracy can be obtained [22–24,15]. Similar to modal analysis, the subspace integration method presented in [25] reduces the degrees of freedom (DOFs) of the simulated objects using some precomputed deformation basis.

To achieve high performance, a linear Cauchy strain model is usually employed that gives a constant stiffness matrix. However it usually gives severe deformation artifacts when the simulated object undergoes a large rotational motion away from its rest state. Muller et al. suppress these artifacts with a warped stiffness approach by tracking the rotational motion of the vertices and cells [7,13]. Choi et al. [14] take an approach similar in sprit to the warped stiffness in their extended modal analysis formulation.

Recently, James et al. [10] proposed a data driven approaches to interactive dynamic simulation. These approaches basically precompute the Green's function, and model the deformation as a boundary value problem in terms of the precomputed Green functions. And it has been shown that the discrete Green functions of real world objects can be computed from some measured quantities [26]. Our approach also derives a deformation model using a precomputed force-displacement matrix. But, we model the interior of the solid objects as well as the boundary, which differentiates our work from those of James et al. [27, 10,28]. And we compress the force-displacement matrix using the CPCA method, while James et al. [28] deals with it using wavelet decomposition based on a multiresolution representation of the object's boundary surface. The advantage of the CPCA based compression method is that it does not require any special (multiresolution) structure on the mesh of the object undergoing deformation.

3. Overview

In this section, we will give the basic formulations for physically based simulation. Let $\Omega \subset R^3$ be a solid shape to be deformed, and $\mathbf{p}(\mathbf{x},t) : \Omega \times R \rightarrow R^3$ be the time dependent motion function of the shape. We represent the motion function $\mathbf{p}(\mathbf{x},t)$ as the sum of the rest state and a displacement $\mathbf{q}(\mathbf{x},t)$:

$$\mathbf{p}(\mathbf{x},t) = \mathbf{x} + \mathbf{q}(\mathbf{x},t).$$

In the finite element method, the shape domain Ω is divided into elements of finite size. And the continuous displacement field in each element is interpolated using the displacement $\mathbf{q}_i(t)$ on the nodal points and a set of piecewise linear basis functions $\{\phi^i(\mathbf{x}) : i = 1,2,\ldots,n\}$ on Ω:

$$\mathbf{q}(\mathbf{x},t) = \mathbf{q}_i(t))\phi^i(\mathbf{x}).$$

Let $\mathbf{q}(t) = [\mathbf{q}_1(t),\ldots,\mathbf{q}_n(t)]^t$ be the time dependent displacement vector. Then, the Euler-Lagrange motion equation for the deformation dynamics becomes:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}_{ext} + \mathbf{f}_{\psi},$$

where $\mathbf{M},\mathbf{D},\mathbf{K}$ are the mass, damping, and stiffness matrices of the simulation system, $\mathbf{f}_{ext}$ is the external force added to the deformable object, and $\mathbf{f}_{\psi}$ is the constraint force added on the constrained node set ψ. In our system, the constrained nodes set ψ is dynamically determined during the simulation according to user manipulation.

Let h be the time step for simulation, and $\Delta\dot{\mathbf{q}} = \dot{\mathbf{q}}(t+h) - \dot{\mathbf{q}}(t)$. We use the implicit Euler method to solve the above dynamic equation, yielding the following linear system:

$$(\mathbf{M} + h\mathbf{D} + h^2\mathbf{K})\Delta\dot{\mathbf{q}} = h(\mathbf{f}_{ext} - \mathbf{D}\dot{\mathbf{q}} - \mathbf{K}(\mathbf{q} + h\dot{\mathbf{q}}) + \mathbf{f}_{\psi}). \tag{1}$$

After solving the above linear system, the displacement vector at the next time step can be easily obtained by:

$$\mathbf{q}(t+h) = \mathbf{q}(t) + h(\dot{\mathbf{q}} + \Delta\dot{\mathbf{q}}).$$

For simplicity, we rewrite (1) as follows:

$$\mathbf{A}\Delta\dot{\mathbf{q}} = \mathbf{f}, \tag{2}$$

where $\mathbf{A} = \mathbf{M} + h\mathbf{D} + h^2\mathbf{K}$, and $\mathbf{f} = h(\mathbf{f}_{ext} - \mathbf{D}\dot{q} - \mathbf{K}(\mathbf{q} + h\dot{\mathbf{q}}) + \mathbf{f}_{\psi})$. Note that the mass matrix $\mathbf{M}$ and the damping matrix $\mathbf{D}$ are constant matrices, while the stiffness matrix $\mathbf{K}$ changes with $\mathbf{q}$, which makes $\mathbf{A}$ change accordingly and imposes a huge amount of computational cost on the simulation. Therefore, most previous works use a linear elastic model by restricting the deformation to only relatively small local deformations, which gives a constant stiffness matrix $\mathbf{K}$ and a corresponding constant matrix $\mathbf{A}$.

The above approach using the linear elasticity model is only applicable to small deformations near the rest state regarding a global rotation. Otherwise, there will be noticeable deformation exaggeration, especially when the object undergoes a global rotation. As long as the local deformation is small, it has been shown that such error due to the global rotation can be handled by tracking the global rotation and formulating the deformation in the object's local/reference coordinate frame. Terzopoulos and Witkin proposed such a method in [29] to fulfill the linear elastic equation. If the object is not rotating rapidly, ignoring the motion of rigid reference will not lead to large error. Let $R_{3\times3}$ be the estimated global rotation matrix (see [7]) and

$$\mathbf{R} = \begin{pmatrix} R_{3\times3} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R_{3\times3} \end{pmatrix}_{3n\times3n}.$$

Then, after transforming the force and displacement quantities into the local coordinate frame, the global large deformation can be easily simulated using:

$$\mathbf{R}\mathbf{A}\mathbf{R}^t\Delta\dot{\mathbf{q}} = \mathbf{f}, \quad \Delta\dot{\mathbf{q}} = \mathbf{R}\mathbf{A}^{-1}\mathbf{R}^t\mathbf{f}. \tag{3}$$

Note that each 3×3 block element $(g_{ij})_{3\times3}$ of $\mathbf{A}^{-1}$ gives the i-th node's response in terms of displacement acceleration to the force acting on the j-th note for a time period of h. Therefore, we intuitively call $\mathbf{A}^{-1}$ the force-displacement matrix of the deformation system and denote $\mathbf{G} \equiv \mathbf{A}^{-1}$, where each of the three column vectors of $\mathbf{G}$ is called the force-displacement vector of the corresponding nodes.

4. Domain Decomposition Method

The above basic approach is not suitable for an object with long and soft components, since there does not exist such a global rotation matrix, as shown in Figure 1. For such objects, we propose to partition the whole object into simple sub-objects and solve the deformations using Domain Decomposition Methods (DDM). DDM has been extensively studied in applied mathematics and mechanical engineering for partial differential equations (PDEs) in the last two decades [30], which is indeed a basic concept of numerical methods for PDEs in general. The principle of DDM is to split the original domain of computation into smaller simpler sub-domains, compute local simplified solutions, and use efficient algebraic solvers to properly interface these solutions.

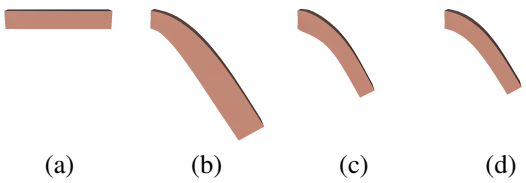


Figure 1. Deformation results with a linear elasticity model. The bar shown in (a) is deformed under the gravity with one end fixed. (b) Results with exaggeration artifacts; (c) Improved result by tracking a global rotational motion; (d) Further improved result produced by using our DDM based method.

Suppose that the object Ω is partitioned into m non-overlapping sub-objects $\Omega = \Omega_1 \cup \ldots \cup \Omega_m$. The shared nodes between neighboring sub-objects are duplicated. Under the DDM framework, each sub-object Ω_i is modeled independently, yielding a system matrix $\mathbf{A}_i$, and a force-displacement matrix $\mathbf{G}_i$. According to Eq. (3), we have

$$\mathbf{R}_i\mathbf{A}_i\mathbf{R}_i^t\Delta\dot{\mathbf{q}}_i = \mathbf{f}_i, \quad \Delta\dot{\mathbf{q}}_i = \mathbf{R}_i\mathbf{G}_i\mathbf{R}_i^t\mathbf{f}_i, \tag{4}$$

where $\mathbf{R}_i$ is the global rotation matrix of sub-object Ω_i being tracked during simulation, and $\mathbf{q}_i$ and $\mathbf{f}_i$ are respectively the displacement vector and force vector

of Ω_i. Denote $\mathbf{H}_i = \mathbf{R}_i\mathbf{A}_i\mathbf{R}_i^t$, $\mathbf{J}_i = \mathbf{R}_i\mathbf{G}_i\mathbf{R}_i^t$, and

$$\breve{\mathbf{q}} = \begin{pmatrix} \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_m \end{pmatrix}, \quad \Delta\breve{\dot{\mathbf{q}}} = \begin{pmatrix} \Delta\dot{\mathbf{q}}_1 \\ \vdots \\ \Delta\dot{\mathbf{q}}_m \end{pmatrix}, \quad \breve{\mathbf{f}} = \begin{pmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_m \end{pmatrix},$$

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{H}_m \end{pmatrix}, \quad \mathbf{J} = \begin{pmatrix} \mathbf{J}_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{J}_m \end{pmatrix}.$$

Then, we have $\mathbf{H}\Delta\breve{\dot{\mathbf{q}}} = \breve{\mathbf{f}}$ and $\mathbf{H}^{-1} = \mathbf{J}$.

Recall that there are some duplicate nodes between the neighboring sub-objects, which should have the same displacement quantities to maintain geometry continuity. This can be accomplished by introducing a vector of Lagrange multipliers λ to enforce a group of point-to-point constraints [31]. Therefore, we have following equations:

$$\mathbf{B}\Delta\breve{\dot{\mathbf{q}}} = \mathbf{c},$$

where $\mathbf{c}$ is a vector of zero, and $\mathbf{B}$ is a matrix of 3×3 blocks. Each row of $\mathbf{B}$ corresponds to a pair of duplicated nodes. Let the k-th row of $\mathbf{B}$ correspond to the i-th and j-th node in $\breve{\mathbf{q}}$, then $\mathbf{B}_{ki} = -\mathbf{B}_{kj} = I_{3\times3}$, and $\mathbf{B}_{kl} = \mathbf{0}$ for $l \neq i, j$. Then, we have the following linear system for each simulation step in our domain decomposition method:

$$\begin{pmatrix} \mathbf{H} & \mathbf{B}^t \\ \mathbf{B} & 0 \end{pmatrix} \begin{pmatrix} \Delta\breve{\dot{\mathbf{q}}} \\ \lambda \end{pmatrix} = \begin{pmatrix} \breve{\mathbf{f}} \\ \mathbf{c} \end{pmatrix}. \tag{5}$$

Note that other linear constraints for user manipulation can be easily formulated as above by appending more rows to $\mathbf{B}$, and some values to $\mathbf{c}$.

Applying the elimination method, the solution of Eq. (5) can be obtained by:

$$\begin{array}{rcl} \Delta\dot{\mathbf{q}}_i & = & \mathbf{R}_i\mathbf{A}_i^{-1}\mathbf{R}_i^t(\mathbf{f}_i - \mathbf{B}_i^t\lambda), \\ \mathbf{W}\lambda & = & \sum_i \mathbf{B}_i\mathbf{R}_i\mathbf{A}_i^{-1}\mathbf{R}_i^t\mathbf{f}_i - \mathbf{c}, \end{array} \tag{6}$$

where $\mathbf{W} = \sum_i \mathbf{B}_i\mathbf{R}_i\mathbf{A}_i^{-1}\mathbf{R}_i^t\mathbf{B}_i^t$. Each simulation step mainly requires the calculation of the following quantities:

a) $\mathbf{R}_i^t\mathbf{f}_i$, $\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{f}_i)$, and $\mathbf{R}_i(\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{f}_i))$;
b) $\mathbf{R}_i^t\mathbf{B}_i^t$, $\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{B}_i^t)$, and $\mathbf{B}_i\mathbf{R}_i(\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{B}_i^t))$;
c) λ and $\Delta\dot{\mathbf{q}}_i$.

The quantities involving $\mathbf{A}_i^{-1}$ can be formed by solving some linear systems with matrix $\mathbf{A}_i$. Since $\mathbf{A}_i^{-1}$ is sparse and symmetric in our problem, some iterative algorithms, such as the conjugate gradient (CG) solver, could be used. Since there are quite number of quantities involving $\mathbf{A}_i^{-1}$ (each pair of shared boundary nodes correspond to one), the naive CG-based domain decomposition method will be extremely slow for a modest scale of simulated object, and cannot meet interactive simulation requirements. In the following sections, we present two novel methods to develop an efficient deformation system.

5. DDM Solvers

5.1. Clustering Method

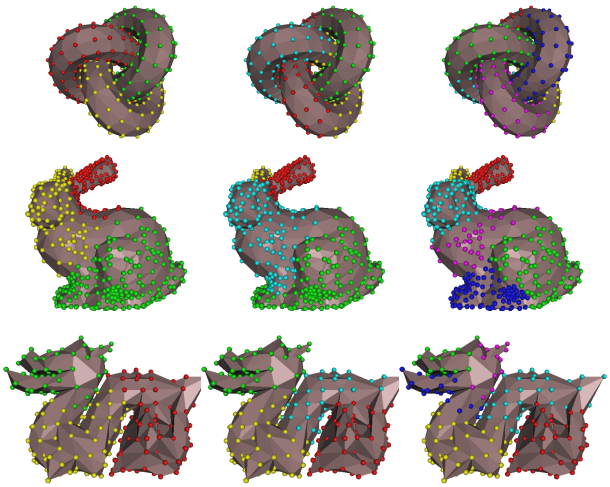
The first method is based on the observation that if the force-displacement matrix $\mathbf{G}_i$ for each subdomain is available, then the solution of the linear systems with matrix $\mathbf{A}_i$ can be obtained simply by matrix-vector multiplications. However, the force-displacement matrix $\mathbf{G}_i$ is usually dense, which may require a huge amount of memory and flops doing the matrix-vector multiplication, yielding an even slower performance. Therefore it is necessary to first compress the force-displacement matrix. We note that there have been considerable efforts in directly approximating an inverse matrix using a sparse matrix [32–34]. However they are developed to obtain better pre-conditioners for fast convergence.

Recall that the force-displacement matrix gives the object's response in terms of displacement acceleration to forces acting on a vertex. Nearby vertices may have a similar response to the same forces, i.e., there exists spatial coherence among the row vectors of the force-displacement matrix. Based on this observation, we propose to compress the force-displacement matrix using clustered principal component analysis [12]. By the CPCA method, the row vectors of $\mathbf{G}_i$ are grouped into several clusters, and each cluster is then approximated using the conventional principal component analysis method. Therefore, the approximation to the force-displacement matrix takes the following form (the subscript i for the sub-domain is omitted for simplicity):

$$\mathbf{G} \approx \begin{pmatrix} \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^t \\ \cdots \\ \mathbf{U}_k\mathbf{S}_k\mathbf{V}_k^t \end{pmatrix} \equiv \bar{\mathbf{G}}, \tag{7}$$

where k is the cluster number, $\mathbf{U}_j$ and $\mathbf{V}_j$ are the left

and right eigen vectors of the j-th cluster, and $\mathbf{S}_j$ are the j-th cluster's eigen values.



3 clusters 4 clusters 6 cluseter

Figure 2. False color visualization of the clusters generated by the CPCA method. Each cluster is indicated by the same color.

Figure 2 shows the false color visualization of clusters obtained by our clustering method. It is very interesting that the clusters have spatial coherences and reflect the shape structures. Table 1 shows the matrix approximation errors of the clustering method for the Torus-Knot model and the Bunny model using different number of clusters and different number of average float elements per node (by choosing the number of the eigen vectors). The results verify that as more clusters and principal components are used, a more accurate approximation is obtained.

There are two practical issues with our clustering method. One issue regards the dimension size of $\mathbf{G}$. For an object with 10K nodes, $\mathbf{G}$ is a 30K$\times$30K matrix, and uses 3.6GB storage. It is obviously not practical to directly run the CPCA method on such a huge matrix. So, it is necessary to reduce the dimension of the row vectors before applying the CPCA method. Recalling the physical meaning of the force-displacement matrix, a vertex also should have similar responses to the forces acting on nearby vertices. Therefore, we first do another node clustering using the node connectivity and coordinates to simplify the mesh of the object down to a desired vertex number. Based on this node clustering, we then sum up the corresponding columns in the force-displacement matrix $\mathbf{G}$, yielding some dimension reduced row vectors. We then apply the CPCA method to group the dimension reduced vectors into some clusters. Note that the resulting clusters are also applicable to the original vectors. So, we group the original vectors into clusters accordingly, and perform a SVD step for each cluster.

Bunny				
flp†	20	60	100	140
1 cluster	708.42 0.2364	170.67 0.1493	125.39 0.1311	105.70 0.1205
3 clusters	1616.9 0.2476	154.77 0.1310	113.38 0.1223	95.364 0.1114
6 clusters	2894.8 0.3377	210.59 0.1461	136.73 0.1270	106.91 0.1136
Torus-Knot				
flp†	20	60	100	140
1 cluster	1315.9 0.2796	87.757 0.0821	48.688 0.0659	39.225 0.0592
3 clusters	1252.7 0.2212	84.452 0.0758	43.710 0.0608	35.658 0.0546
6 clusters	1604.7 0.2472	103.48 0.0733	48.139 0.0610	35.881 0.0539
flp† denotes the average float elements per node.				

Table 1
Approximation error of the compressed force-displacement matrices using 1, 3 and 6 clusters, and 20, 40, 100 and 140 float elements per node. In each cell, the top value is $||\mathbf{G}-\bar{\mathbf{G}}||_F$, and the bottom value is the sum of the eigen values of $\mathbf{G}-\bar{\mathbf{G}}$.

The other issue is how to invert $\mathbf{A}$ to obtain the force-displacement matrix $\mathbf{G}$. It is still an open mathematical problem to efficiently invert a huge matrix. In our current implementation, we first compute the Choleschy factorization of $\mathbf{A}$, then compute $\mathbf{G}$'s j-th column $\mathbf{g}_j$ by solving $\mathbf{A}\mathbf{g}_j = \mathbf{e}_j$, where $\mathbf{e}_j$ is the j-th canonical basis.

With the compressed force-displacement matrix, the solution to the linear system of $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be

simply computed by:

$$\mathbf{x} = \mathbf{G}\mathbf{b} \approx \begin{pmatrix} \mathbf{U}_1\mathbf{S}_1(\mathbf{V}_1^t\mathbf{b}) \\ \cdots \\ \mathbf{U}_k\mathbf{S}_k(\mathbf{V}_k^t\mathbf{b}) \end{pmatrix}.$$

Now we analyze the compression ratio and performance gain in solving $\mathbf{A}\mathbf{x} = \mathbf{b}$. For simplicity, we suppose that the clusters have the same number m of principal components. Then the compressed force-displacement matrix takes about $(1+k)\cdot m\cdot|\mathbf{G}|$ elements, where $|\mathbf{G}|$ denotes the size of the $\mathbf{G}$. Therefore the storage compression ratio is about $\frac{(1+k)\cdot m}{|\mathbf{G}|}$. Note that the computational cost is proportional to the element number in the matrix representation, so the computational cost ratio is also about $\frac{(1+k)\cdot m}{|\mathbf{G}|}$. As long as $(1+k)\cdot m \ll |\mathbf{G}|$, our clustering method greatly reduces the storage and increases performance, compared with using an uncompressed force-displacement matrix $\mathbf{G}$. Similar analysis can be done when the clusters have different numbers of principal components.

When $\mathbf{b}$ is sparse, the solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be formed even faster, which demonstrates another advantage of our clustering method. And in our domain decomposition framework, most of the linear systems have $\mathbf{R}_i^t\mathbf{B}_i^t$ as the right hand side vector, which are extremely sparse, as shown in Section 4.

5.2. Hybrid Method

The second method is a hybrid that takes advantage of both the force-displacement matrix and the Cholesky factorization of $\mathbf{A}_i$. The basic ideas behind the hybrid method are: (1) compute and reuse the Cholesky factorization of $\mathbf{A}_i$ to solve the linear systems with matrix $\mathbf{A}_i$; (2) compute and reuse the force-displacement vectors of the boundary nodes for on-the-fly calculation of $\mathbf{B}_i\mathbf{R}_i(\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{B}_i^t))$.

To help understand the hybrid method, let us consider solving a linear system $\mathbf{A}_i\mathbf{x} = \mathbf{b}$. We can solve it either by using the compressed force-displacement matrix $\bar{\mathbf{G}}_i$, or using the Cholesky factorization of $\mathbf{A}_i$. The performance of using $\bar{\mathbf{G}}_i$ depends on the number of the principal components used. In our experiments, it is usually similar to the performance of using factorized $\mathbf{A}_i$. So this leads us to idea (1). Though the performances are similar, we can use factorized $\mathbf{A}_i$ to get rid of the heavy pre-computation stage to calculate and compress the whole force-displacement matrix. In our experiments, we can compute the Cholesky factorization on the fly using the UMFPack [35].

However, if $\mathbf{b}$ is very sparse, e.g., $\mathbf{b}$ is a column of $\mathbf{R}_i^t\mathbf{B}_i^t$, then the performance of using factorized $\mathbf{A}_i$ will be several times slower than that of using $\bar{\mathbf{G}}_i$. This motivates us to incorporate idea (2). We also compute the force-displacement vector for the shared boundary nodes on-the-fly using factorized $\mathbf{A}_i$ beforehand. In addition, for each boundary node, only a small part of the force-displacement vector will be used when calculating $\mathbf{B}_i\mathbf{R}_i(\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{B}_i^t))$ in Section 4. So we need store only a very small portion of the force-displacement vectors for the boundary nodes in each cluster. This further reduces the memory cost and computation cost.

In the clustering method, some eigen vectors in $\mathbf{A}_i^{-1}$ are discarded for compression, which sometimes produces a singular matrix $\mathbf{W}$. To avoid the possible instability, we have to do expensive singular value decomposition on $\mathbf{W}$ to obtain a numerically stable result for the Lagrange multiplier λ. But, in the hybrid method, $\mathbf{W}$ is always of full rank, so that λ can be solved with standard efficient routines, e.g., dpovs provided in LAPACK.

Now we analyze the storage cost and computational cost of the hybrid method for the i-th sub domain. The storage consists of two major parts. One is for the Cholesky factorization of $\mathbf{A}_i$, which is propositional to the number of the non-zero elements in $\mathbf{A}_i$, which is itself proportional to the number of nodes in the sub-domain. The other is for the force-displacement vectors, which is proportional to the square of the boundary node number in the sub-domain, which is usually quite small.

The computational cost consists of three parts. One part is for solving linear systems with factorized $\mathbf{A}_i$, which is proportional to the node number in the sub-domain. Another is for calculating $\mathbf{B}_i\mathbf{R}_i(\mathbf{A}_i^{-1}(\mathbf{R}_i^t\mathbf{B}_i^t))$. Since the involved elements in $\mathbf{A}_i^{-1}$ have been pre-computed for reuse, this part of the cost is proportional to the square of the boundary node number in the sub-domain. The last part is for solving the Lagrange multipliers λ for all subdomains with matrix $\mathbf{W}$. Since $\mathbf{W}$ is a dense matrix, this part of the computation cost is $O\left(|\mathbf{W}|^3\right)$.

6. Experiment Results

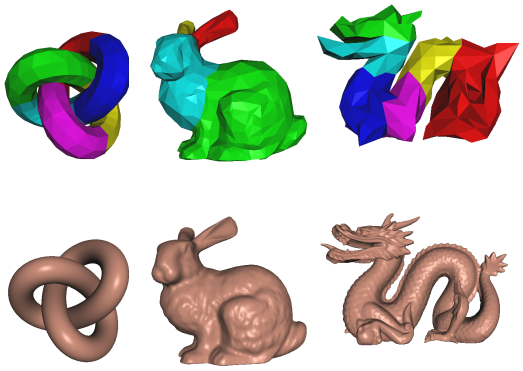


Figure 3. Some models used in our experiments. The top row shows a tetrahedral mesh with colors indicating the sub-domains, while the bottom row shows a surface mesh used for display.

In this section, we will present some experimental results achieved with our deformation system. First, we show some models used in the experiments in Figure 3. In our system, the simulation is run on a relatively coarser tetrahedron mesh, which is then used to deform finer surface meshes by piecewise linear interpolation for display. For each vertex of the surface mesh, we first find the tetrahedron that contains it, or the closest one if the vertex lies out of the tetrahedral mesh; then we compute the vertex's barycentric coordinates in the tetrahedron for runtime interpolation when the object is deformed. The statistic numbers of these models together with the performances are listed in Table 2. The timing was obtained on a PC with a Pentium IV CPU and 1 GB RAM. Our hybrid method can run at about 40 $\sim$ 100 fps on these models, which is two times faster than the clustering method. Interpolating the surface mesh requires some time, but it can be easily accelerated with graphics hardware.

Figure 4 shows some deformation results on the Bunny model, altered by a user. Figure 5 shows some frames of a simulation sequence of the Dragon model,

	Bar	Torus	Bunny	Dragon
#Elem	2205	1354	2436	843
#Node	611	517	878	343
#NSub	3	6	4	6
#BNode	50	52	66	55
#VSurface	-	32768	34835	101108
simulation time (second/step)				
CG-Based	1.010	0.580	3.980	0.180
Clustering	0.035	0.030	0.064	0.020
Hybrid	0.019	0.012	0.026	0.010

Table 2
Model sizes and performance timing. #Elem and #Node denote the tetrahedron number and node number of the tetrahedral mesh, #NSub denotes the subdomain number, #BNode denotes the boundary node number, and #VSurface denotes the surface mesh vertex number. The rows of CG-Based, Clustering and Hybrid show simulation time of the naive CG based DDM, our clustering method, and the hybrid method.

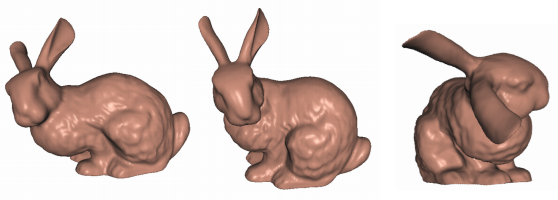


Figure 4. Simulation results of the Bunny model.

which falls due to gravity and bounces on the ground plane. In Figure 6, we show two extremely large deformation results of the Dragon model achieved in our system.

At last, we compare the simulation results of our method with DDM and without DDM in Figure 7. This figure clearly shows that the results without DDM suffer from severe distortions and exaggerations, especially in the area highlighted with a rectangle, while our DDM produces natural deformation results.

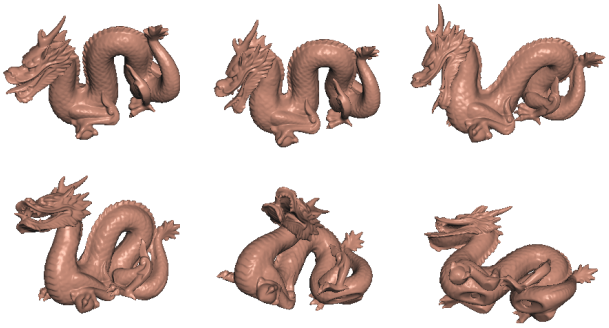


Figure 5. Simulation results of the Dragon model freely bouncing on a ground plane.

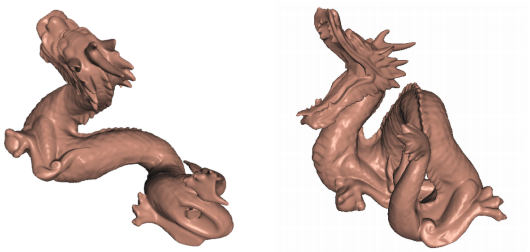


Figure 6. Large deformation results of the Dragon model.

7. Discussion

We have presented a physically based deformation method using the precomputed force-displacement matrix. The advantages of our method are

- It is very fast. The fast performance is achieved by precomputing a force-displacement matrix.
- It is stable, and allows for relatively large time steps, since essentially the implicit Euler method is used to solve the underlying differential equations.
- Large deformations are handled very well by tracking global rotations and using domain decomposition methods. DDM improves the result more than with the linear blend described in [15]. And compared with [7,8], our algorithm can utilize more precomputed information.
- It allows for dynamically introducing new constraints for user manipulations.

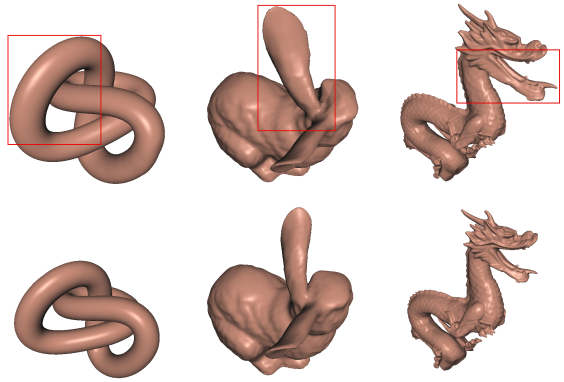
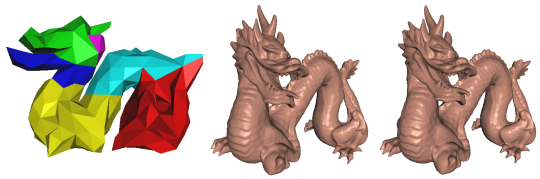


Figure 7. Result comparison between tracking a single global rotation method (top row) and our domain decomposition method (bottom row). There are noticeable distorted areas highlighted with a rectangle in the top row.

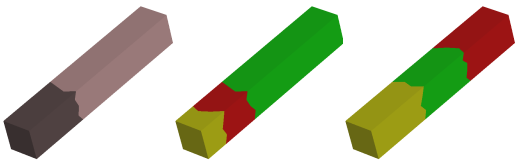
An open problem with the domain decomposition method is how to partition the simulated object into sub-domains. Recall that we assume the sub-domains will not undergo large local deformations, and the partition boundary needs to be as small as possible for better performance. So, it is desirable to partition the objects at the joints and the concave regions, e.g., the root of the ear of the Bunny model. It is also desirable to partition the object at the softest regions, if the stiffness parameter varies over the simulated object. In our current system, the partitions are manually done by a user. And we encountered some difficulties with the manual partitioning of complex geometric shapes, e.g., the Torus-Knot model and the Dragon model.

Therefore, it is necessary and worthwhile to investigate some automatical partitioning methods. Recall that nodes with similar force-displacement vectors will have similar movement acceleration, so the



(a) (b) (c)

Figure 8. Automatic partition of the Dragon model using the CPCA method. (a) Parition result with 6 sub-domains; (b) A simulation result with the automatically partitioned sub-domains; (c) Another simulation result using 6 manually partitioned sub-domains.



(a) (b) (c)

Figure 9. Automatic partition of a bar with spatially varying stiffness paramters. (a) False color visualization of the stiffness with brighter colors indicating greater stiffness; (b) Partition result on (a); (c) Another partitioning result on the same bar but with uniform stiffness parameters.

CPCA clusters could give a reasonable partition of the simulated object for the DDM method. In order to run CPCA method over the tetrahedron cells, we first assign each tetrahedron cell with the sum of the force-displacement vectors of its four nodes. Figure 8(a) shows an automatic partitioning of the Dragon model. The corresponding deformation result shown in Figure 8(b) is encouraging, compared with the result with manual partition shown in Figure 8(c). Figure 9 shows an automatic partitioning result on a bar with spatially varying stiffness parameters. The results show that the CPCA based partition method can automatically put more clusters in the soft part of the bar. These experimental results suggest an interesting future work on automatic domain decomposition.

8. Acknowledgments

We would like to thank Dr. Steve Lin in MSRA for proofreading the paper and providing many useful comments. This project is partially supported by Natural Science Foundation of China under Grant No. 60021201, The National Basic Research Program of China (973 Program) under Grant No. 2002CB312100, and Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China under Grant No. 705027.

REFERENCES

1. J. E. Chadwick, D. R. Haumann, R. E. Parent, Layered construction for deformable animated characters, in: Proceedings of the 16th annual conference on Computer graphics and interactive techniques, ACM Press, 1989, pp. 243–252.
2. X. Tu, D. Terzopoulos, Artificial fishes: physics, locomotion, perception, behavior, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM Press, 1994, pp. 43–50.
3. D. Baraff, A. Witkin, Large steps in cloth simulation, in: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM Press, 1998, pp. 43–54.
4. K.-J. Choi, H.-S. Ko, Stable but responsive cloth, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 604–611.
5. R. Bridson, S. Marino, R. Fedkiw, Simulation of clothing with folds and wrinkles, in: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, 2003, pp. 28–36.
6. R. D. Cook, D. S. Malkus, M. E. Plesha, Concepts and Applications of Finite Element Analysis, 3rd Edition, John Wiley & Sons, 1989.
7. M. Müller, J. Dorsey, L. McMillan, R. Jagnow, B. Cutler, Stable real-time deformations, in: Proceedings of the 2002 ACM SIG-

GRAPH/Eurographics symposium on Computer animation, ACM Press, 2002, pp. 49–54.
8. M. Müller, M. Gross, Interactive virtual materials, in: GI '04: Proceedings of the 2004 conference on Graphics interface, Canadian Human-Computer Communications Society, 2004, pp. 239–246.
9. S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popovic, Interactive skeleton-driven dynamic deformations, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 586–593.
10. D. L. James, D. K. Pai, Real time simulation of multizone elastokinematic models, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2002, pp. 927–932.
11. J. Huang, X. Liu, H. Bao, B. Guo, H.-Y. Shum, Clustering method for fast deformation with constraints, in: SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling, ACM Press, New York, NY, USA, 2005, pp. 221–226.
12. P.-P. Sloan, J. Hall, J. Hart, J. Snyder, Clustered principal components for precomputed radiance transfer, ACM Trans. Graph. 22 (3) (2003) 382–391.
13. M. Müller, M. H. Gross, Interactive virtual materials., in: Graphics Interface, 2004, pp. 239–246.
14. M. G. Choi, H.-S. Ko, Modal warping: Real-time simulation of large rotational deformation and manipulation, IEEE Trans. Vis. Comput. Graph. 11 (1) (2005) 91–101.
15. K. Hauser, C. Shen, J. F. O'Brien, Interactive deformations using modal analysis with constraints, in: Proceedings of Graphics Interface 2003, 2003, pp. 247—256.
16. S. F. F. Gibson, B. Mirtich, A survey of deformable modeling in computer graphics, Tech. Rep. TR-97-19, Mitsubish Electric Research Lab., Cambridge (November 1997).
17. A. Nealen, M. Müller, R. Keiser, E. Boserman, M. Carlson, Physically based deformable models in computer graphics, in: Eurographics 2005 state of the art report (STAR), 2005.
18. S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popovic, A multiresolution framework for dynamic deformations, in: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM Press, 2002, pp. 41–47.
19. E. Grinspun, P. Krysl, P. Schröder, Charms: a simple framework for adaptive simulation, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 281–290.
20. X. Wu, M. S. Downes, T. Goktekin, F. Tendick, Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes, Comput Graphics Forum 20 (3) (2001) 349–358.
21. G. Debunne, M. Desbrun, M.-P. Cani, A. H. Barr, Dynamic real-time deformations using space & time adaptive sampling, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM Press, 2001, pp. 31–36.
22. A. Pentland, J. Williams, Good vibrations: model dynamics for graphics and animation, in: Proceedings of the 16th annual conference on Computer graphics and interactive techniques, ACM Press, 1989, pp. 215–222.
23. J. Stam, Stochastic dynamics: Simulating the effects of turbulence on flexible structures, Comput Graphics Forum 16 (3) (1997) 159–164.
24. D. L. James, D. K. Pai, Dyrt: dynamic response textures for real time deformation simulation with graphics hardware, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM Press, 2002, pp. 582–585.
25. J. Barbič, D. L. James, Real-time subspace integration for st. venant-kirchhoff deformable models, ACM Trans. Graph. 24 (3) (2005) 982–990.
26. D. K. Pai, K. van den Doel, D. L. James, J. Lang, J. E. Lloyd, J. L. Richmond, S. H. Yau, Scanning physical interaction behavior of 3d objects, in: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM Press, 2001, pp. 87–96.
27. D. L. James, D. K. Pai, Artdefo: accurate real time deformable objects, in: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 65–72.
28. D. L. James, K. Fatahalian, Precomputing interactive dynamic deformable scenes, ACM Trans.

Graph. 22 (3) (2003) 879–887.
29. D. Terzopoulos, A. Witkin, Physically based models with rigid and deformable components, IEEE Comput. Graph. Appl. 8 (6) (1988) 41–51.
30. A. Quarteroni, A. Valli, Domain Decomposition Methods for Partial Differential Equations, Oxford Science Publications, Clarendon Press, Oxford, 1999.
31. D. Metaxas, D. Terzopoulos, Dynamic deformation of solid primitives with constraints, Computer Graphics (Proc. SIGGRAPH'92) 26 (2) (1992) 309–312.
32. M. Benzi, M. Tuma, A comparative study of sparse approximate inverse preconditioners, Applied Numerical Mathematics: Transactions of IMACS 30 (2–3) (1999) 305–340.
33. M. Benzi, J. K. Cullum, M. Tuma, Robust approximate inverse preconditioning for the conjugate gradient method, SIAM Journal on Scientific Computing 22 (4) (2000) 1318–1332.
34. R. Bridson, W.-P. Tang, Multiresolution approximate inverse preconditioners, SIAM Journal on Scientific Computing 23 (2) (2001) 463–479.
35. T. A. Davis, Umfpack version 4.1 user guide, Tech. Rep. TR-03-008, University of Florida (2003).