Computers & Graphics (2017)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

As-rigid-as-possible solid simulation with oriented particles

Min Gyu Choi[a,*], Jehee Lee[b]

[a] Department of Computer Science, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul 01897, Korea
[b] Department of Computer Science and Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea

ARTICLE INFO

Article history:
Received 15 June 2017
Revised 19 July 2017
Accepted 19 July 2017

Keywords:
Physics-based simulation
Dynamic deformation
Deformation graph
Oriented particle

ABSTRACT

We propose a new as-rigid-as-possible approach to the real-time simulation of physics-based deformable models for interactive applications such as computer games. The key observation is that the efficacy of an embedded oriented particle representation and the stability of a variational implicit formulation of the projective dynamics are complementary to each other. We reformulate the variational implicit formulation to deal with an embedded graph of oriented particles. Our new formulation is extremely stable, and our alternating local/global optimization solver is both easy to implement and computationally efficient. Our method can deal with one-dimensional (cable and rod), two-dimensional (shell), and three-dimensional (solid) models in a uniform manner. Experimental results demonstrate that hundreds of deformable models with an extremely large number of polygons can be simulated robustly in real time using thousands of particles.

1. Introduction

Physics-based simulation is becoming one of the most important techniques for interactive applications such as games. In this paper, we consider physics-based deformation, which has been studied extensively in computer graphics. However, it is still challenging to robustly produce visually convincing and stable deformation in real time. In real-time applications, the robustness and computational efficiency of deformation simulations are often more important than their accuracy.

Recently, several simulation techniques have been developed to compute positions directly instead of integrating velocities or accelerations to achieve better stability and efficiency. Position-based dynamics (PBD) [1] deals directly with mesh vertices, while meshless deformation [2] utilizes shape matching to compute the optimal rotation and translation for mesh vertices. These position-based methods are stable regardless of the time step size, and thus the deformation can be simulated very efficiently with large time steps.

Two remarkable approaches aim to further improve the PBD method. The first approach employs a simplified structure with a small number of oriented particles to simulate the complex geometry of meshes in a more efficient and robust manner [3]. The second approach called projective dynamics [4] reformulates the implicit time integration of deformation dynamics as energy minimization in a variational form. In PBD, the material stiffness of a deformable solid is tightly coupled with the convergence of the solver. The formulation of variational implicit integration is similar to PBD, but the projective dynamics method is advantageous because the material stiffness can be specified independently of the solution methods.

We found that these two approaches are complementary to each other and they can be combined to take advantage of their benefits. The key challenge is to reformulate the deformation energy and momentum potential energy to deal with an embedded graph of oriented particles. Our new formulation yields a compact formula via clever manipulation of the integral energies and it is extremely stable. Furthermore, our alternat-

*Corresponding author: Tel.: +82-2-940-5472; fax: +82-2-909-0998;
e-mail: mgchoi@kw.ac.kr (Min Gyu Choi)

ing local/global optimization solver is easy to implement and very efficient for simulating complex deformable models in real time. The deformable models can be manipulated interactively and the collision between deformable models can be handled efficiently. Our method can deal with one-dimensional (cable and rod), two-dimensional (shell), and three-dimensional (solid) models in a uniform manner.

2. Related work

Physics-based simulations of deformable bodies have been researched for decades in computer graphics since the pioneering work by Terzopoulos et al. [5] and many methods have been developed to produce accurate simulations of various types of deformable objects [6]. However, in real-time applications, the robustness and computational efficiency of deformation simulations are often more important than their accuracy, and these requirements are still demanding.

Recently, a number of methods have been proposed for robust real-time simulations by evolving the positions of the particles using the initial and predicted positions, before updating the velocities based on the positions. Meshless deformation based on shape matching computes the optimal rotation and translation from the initial positions to the predicted positions to deform the mesh vertices [2]. This method is robust regardless of the time step size, and thus it is suitable for real-time applications. However, shape matching with a single transformation is restricted to modest deformation. The extension of this method to a lattice applies shape matching to each set of overlapping lattice points with a fast summation method to generate large deformation [7]. The fast summation approach has been applied to the simulation of hair with chain shape matching [8], as well as being generalized to an irregular structure on a surface mesh with volume preservation [9], and extended to multi-resolution approaches to enhance the convergence [10, 11]. Shape matching with oriented particles is an efficient and robust method for simulating the complex dynamic deformation of one-, two-, and three-dimensional deformable bodies even with a small number of particles in a single framework [3]. A deformable body is approximated by ellipsoidal particles and shape matching is applied iteratively to each shape matching group comprising a particle and its one-ring neighbors. Our method employs this oriented particle representation so hundreds of deformable models with an extremely large number of polygons can be approximated using thousands of particles and simulated robustly in real time, as illustrated in Fig. 7.

Shape matching can be interpreted as a geometric constraint in PBD [1]. PBD employs an iterative mass-weighted projection of constraints in a Gauss–Seidel solver. However, the stiffness is highly dependent on the time step size and the iteration count, which is also problematic in shape matching approaches. Nevertheless, PBD is popular for the real-time simulation of deformable models because of its simplicity and robustness. It has been extended to the simulation of fluids [12], rigid bodies [13], elastic rods [14, 15], and volumetric materials [16], as well as all of them in a unified manner [17]. A comprehensive survey of PBD methods was provided in [18]. Recently, PBD was extended to address the stiffness independently of the time step size and the iteration count based on a compliant constraint formulation [19]. However, the extended PBD still employs Gauss–Seidel iterations, and thus its convergence is slower than that of projective dynamics [4, 20] where the global solver benefits from the pre-factored system matrix. Moreover, its application to oriented particles has not been addressed previously.

Recently, a number of fast and parallel techniques have been proposed by formulating implicit Euler integration as energy minimization [21, 22] and by employing an alternating local/global optimization solver. Liu et al. [20] proposed a local/global solver for mass-spring systems, where the local solver deals with nonlinear terms for direction and the global solver handles stretching. This idea was generalized to other constraints in projective dynamics [4]. A Chebyshev semi-iterative approach was also proposed, which combines the results obtained from previous iterations to achieve better convergence in projective dynamics and PBD [23]. Narain et al. [24] employed the alternating direction method of multipliers (ADMM) for implicit time integration and showed that projective dynamics is a special case of ADMM. Their method also allows nonlinear constitutive models and hard constraints. In addition, Liu et al. [25] interpreted projective dynamics as quasi-Newton optimization and applied the L-BFGS method to accelerate convergence. In these techniques, the stiffness is largely independent of the iteration count and the solution becomes more accurate as the number of iterations increases. Our method is also based on implicit Euler integration formulated as energy minimization, and thus it differs from the original oriented particles approach where the stiffness is highly dependent on the time step size and the iteration count.

In as-rigid-as-possible (ARAP) surface modeling [26], a block coordinate descent method is employed to iteratively minimize the shape deformation energy in alternating local/global optimization steps. This method introduces a local rotation at each vertex to define an ARAP deformation energy by using the squared distances between the locally rotated positions of its neighbors and the actual deformed positions. The auxiliary rotations and the positions of the vertices should be optimized. An optimal rotation at a vertex is computed in parallel with shape matching of its neighboring vertices, which requires the polar decomposition of a shape matching matrix. The optimal positions are computed efficiently by solving a linear system, which depends only on the initial mesh, so it can be pre-factored with a sparse Cholesky decomposition.

Embedded deformation for shape manipulation [27] introduces a deformation graph of nodes corresponding to rigid transformations that deform nearby space, and then defines an ARAP deformation energy over the deformation graph. In contrast to ARAP surface modeling, this approach can deal with a wide range of shape representations, such as meshes, polygon soups, mesh animations, and animated particle systems. However, the node has no volume, and thus it cannot deal with the deformation of a one-dimensional structure robustly, unlike the oriented particles approach and our proposed method. In addition, the iterative Newton–Gauss method employed for nonlinear optimization requires more time than the iterative lo-

cal/global optimization method when only seeking a plausible solution in a small number of iterations.

3. Method

In this paper, we introduce a deformation graph $\mathcal{G}$ comprising oriented particles, which approximate a flexible body and deform nearby spaces robustly even with a small number of particles. The positions and orientations of the oriented particles are computed stably by solving an energy minimization problem formulated as implicit Euler integration in a variational form. The total energy $E(\mathcal{G})$ comprises the momentum potential energy $E^k(\mathcal{G})$, ARAP deformation energy $E^e(\mathcal{G})$, and direct manipulation constraint energy $E^c(\mathcal{G})$:

$$E(\mathcal{G}) = E^k(\mathcal{G}) + E^e(\mathcal{G}) + E^c(\mathcal{G}). \tag{1}$$

We develop an iterative local/global optimization solver to seek a plausible solution in an efficient and robust manner. The final deformed vertices are obtained by linear blend skinning of the rigid transformations stored in the oriented particles, which can be implemented with GPU skinning.

Our main contribution lies in developing the ARAP deformation energy over a deformation graph comprising oriented particles. Thus, we first explain the deformation graph and the corresponding space deformation in Section 3.1, before deriving the ARAP deformation energy in Section 3.2. We explain the alternating local/global optimization solver before considering other energies to make the explanation clearer. The momentum potential energy and direct manipulation constraint energy are described in Sections 3.3 and 3.4, respectively.

3.1. Deformation graph

Suppose that the j-th ellipsoidal particle $\mathcal{E}_j$ is transformed from the rest position $\bar{\mathbf{x}}_j$ and orientation $\bar{\mathbf{E}}_j$ to the current position $\mathbf{x}_j$ and orientation $\mathbf{E}_j = [\mathbf{e}_j^a|\mathbf{e}_j^b|\mathbf{e}_j^c]$, where $\mathbf{e}_j^a$, $\mathbf{e}_j^b$, $\mathbf{e}_j^c$ are the current unit axes of $\mathcal{E}_j$. We denote the set of particles directly connected to $\mathcal{E}_j$ using $\mathcal{N}_j$. To compute the deformations efficiently from a deformation graph, we employ linear blend skinning, as described by Sumner et al. [27]. The influences of individual oriented particles are blended smoothly so the deformed position $\mathbf{v}_i$ of each visual vertex $\bar{\mathbf{v}}_i$ becomes a weighted sum of its positions after applying rigid transformations of the k-nearest oriented particles:

$$\mathbf{v}_i = \sum_{j\in\mathcal{K}_i} w_j(\bar{\mathbf{v}}_i)\left[\mathbf{R}_j(\bar{\mathbf{v}}_i - \bar{\mathbf{x}}_j) + \mathbf{x}_j\right], \tag{2}$$

where $\mathbf{R}_j = \mathbf{E}_j\bar{\mathbf{E}}_j^{\mathrm{T}}$ is the rotation of the oriented particle $\mathcal{E}_j$ and $\mathcal{K}_i$ denotes the k-nearest particles of $\bar{\mathbf{v}}_i$. The normal vector can also be obtained similarly, i.e., $\mathbf{n}_i = \sum_{j\in\mathcal{K}_i} w_j(\bar{\mathbf{v}}_i)[\mathbf{R}_j\bar{\mathbf{n}}_i]$, but it requires renormalization.

3.2. ARAP deformation energy

To measure the rigidity of deformation, we employ the ARAP deformation energy introduced in [26, 27]. The deformation energy of the whole body is the sum of the local rigidity energy defined at each node $\mathcal{E}_i$ assuming that its neighbors $\mathcal{E}_j$ are under the local rigid transformation of $\mathcal{E}_i$. It measures the squared distances between the positions of $\mathcal{E}_j$ under the local rigid transformation and the actual deformed positions. The node considered in this paper is an ellipsoid that differs from the single vertex described in [26, 27]. Hence, the ARAP deformation energy of a deformation graph $\mathcal{G}$ is defined using integration over the ellipsoids as follows:

$$E^e(\mathcal{G}) = \frac{k}{2}\sum_i\sum_{j\in\mathcal{N}_i}\int_{\mathcal{E}_j}\left\|\mathbf{R}_i(\bar{\mathbf{x}} - \bar{\mathbf{x}}_i) - (\mathbf{x} - \mathbf{x}_i)\right\|^2 d\mathbf{x}, \tag{3}$$

where k is the stiffness constant and $\bar{\mathbf{x}}$ is the undeformed position of a material point $\mathbf{x}$ in the ellipsoid $\mathcal{E}_j$. We note that $\bar{\mathbf{x}}$ is also a function of the integration variable $\mathbf{x}$. Minimizing the ARAP deformation energy is analogous to enforcing the shape matching constraint for the local rigidity. Our method is not intended for physically correct simulations, but instead it is a practical approach for obtaining visually plausible real-time deformations.

The deformation energy $E^e(\mathcal{G})$ needs to be represented in a compact form for minimization. Thus, we first integrate the term E_{ij} analytically:

$$E_{ij} = \int_{\mathcal{E}_j}\left\|\mathbf{R}_i(\bar{\mathbf{x}} - \bar{\mathbf{x}}_i) - (\mathbf{x} - \mathbf{x}_i)\right\|^2 d\mathbf{x}.$$

Expanding E_{ij} with the definition of the vector norm gives

$$\begin{aligned}E_{ij} = &\int_{\mathcal{E}_j}(\bar{\mathbf{x}} - \bar{\mathbf{x}}_i)^{\mathrm{T}}(\bar{\mathbf{x}} - \bar{\mathbf{x}}_i)d\mathbf{x} - 2\int_{\mathcal{E}_j}(\bar{\mathbf{x}} - \bar{\mathbf{x}}_i)^{\mathrm{T}}\mathbf{R}_i^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_i)d\mathbf{x}\\ &+ \int_{\mathcal{E}_j}(\mathbf{x} - \mathbf{x}_i)^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_i)d\mathbf{x}.\end{aligned}$$

We employ the following integral properties of an ellipsoid: $\int_{\mathcal{E}_j}\mathbf{x}d\mathbf{x} = v_j\mathbf{x}_j$, $\int_{\mathcal{E}_j}\mathbf{x}^{\mathrm{T}}\mathbf{x}d\mathbf{x} = tr(\sqrt{\mathbf{A}_j}^{\mathrm{T}}\sqrt{\mathbf{A}_j}) + v_j\mathbf{x}_j^{\mathrm{T}}\mathbf{x}_j$ and $\int_{\mathcal{E}_j}\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{R}_i^{\mathrm{T}}\mathbf{x}d\mathbf{x} = tr(\sqrt{\mathbf{A}_j}^{\mathrm{T}}\mathbf{E}_j^{\mathrm{T}}\mathbf{R}_i\bar{\mathbf{E}}_j\sqrt{\mathbf{A}_j}) + v_j\bar{\mathbf{x}}_j^{\mathrm{T}}\mathbf{R}_i^{\mathrm{T}}\mathbf{x}_j$, where $\mathbf{A}_j = \frac{1}{5}v_j\mathrm{diag}(a_j^2, b_j^2, c_j^2)$ is the moment matrix of the axis-aligned ellipsoid with the volume $v_j = \frac{4}{3}\pi a_j b_j c_j$ and the principal radii a_j, b_j and c_j. Then, E_{ij} can be written as follows:

$$\begin{aligned}E_{ij} = &\, tr(\sqrt{\mathbf{A}_j}^{\mathrm{T}}\sqrt{\mathbf{A}_j}) + v_j(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)^{\mathrm{T}}(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)\\ &- 2tr(\sqrt{\mathbf{A}_j}^{\mathrm{T}}\bar{\mathbf{E}}_j^{\mathrm{T}}\mathbf{R}_i^{\mathrm{T}}\mathbf{E}_j\sqrt{\mathbf{A}_j}) - 2v_j(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)^{\mathrm{T}}\mathbf{R}_i^{\mathrm{T}}(\mathbf{x}_j - \mathbf{x}_i)\\ &+ tr(\sqrt{\mathbf{A}_j}^{\mathrm{T}}\sqrt{\mathbf{A}_j}) + v_j(\mathbf{x}_j - \mathbf{x}_i)^{\mathrm{T}}(\mathbf{x}_j - \mathbf{x}_i).\end{aligned}$$

Factoring the matrix trace terms and dot product terms gives

$$\begin{aligned}E_{ij} = &\, tr([\mathbf{R}_i\bar{\mathbf{E}}_j\sqrt{\mathbf{A}_j} - \mathbf{E}_j\sqrt{\mathbf{A}_j}]^{\mathrm{T}}[\mathbf{R}_i\bar{\mathbf{E}}_j\sqrt{\mathbf{A}_j} - \mathbf{E}_j\sqrt{\mathbf{A}_j}])\\ &+ v_j\left\|\mathbf{R}_i(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i) - (\mathbf{x}_j - \mathbf{x}_i)\right\|^2.\end{aligned}$$

Now, we exploit a property of the Frobenius norm $tr(\mathbf{M}^{\mathrm{T}}\mathbf{M}) = \|\mathbf{M}\|_F^2$ to obtain E_{ij} of the form:

$$E_{ij} = \left\|\mathbf{R}_i\bar{\mathbf{E}}_j\mathbf{A}_j^{\frac{1}{2}} - \mathbf{E}_j\mathbf{A}_j^{\frac{1}{2}}\right\|_F^2 + v_j\left\|\mathbf{R}_i(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i) - (\mathbf{x}_j - \mathbf{x}_i)\right\|^2.$$

Finally, the analytic integration of Eq. (3) yields a discrete version of the ARAP deformation energy:

$$\begin{aligned}E^e(\mathcal{G}) = \frac{k}{2}\sum_i\sum_{j\in\mathcal{N}_i}\Big(&\|\mathbf{R}_i(\bar{\mathbf{E}}_j\mathbf{A}_j^{\frac{1}{2}}) - \mathbf{R}_j(\bar{\mathbf{E}}_j\mathbf{A}_j^{\frac{1}{2}})\|_F^2\\ &+ v_j\|\mathbf{R}_i(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i) - (\mathbf{x}_j - \mathbf{x}_i)\|^2\Big),\end{aligned} \tag{4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

To find the optimal solution that minimizes the energy given in Eq. (4), we employ an alternating local/global optimization approach in a similar manner to that described by Sorkine and Alexa [26]. In the local step, the optimal rotations of the particles are computed independently under the assumption that their positions are fixed. In the global step, the positions are computed simultaneously by solving a linear system under the assumption that the rotations are fixed. The local/global steps are performed repeatedly for a user-specified number of times.

Optimal rotation. To compute the optimal rotation $\mathbf{R}_i$ of the oriented particle $\mathcal{E}_i$, we first assume that all the positions are fixed in Eq. (4). Under this assumption, $\mathbf{R}_i$ becomes independent of $\mathbf{R}_j$ in [26]. However, $\mathbf{R}_i$ is related to $\mathbf{R}_j$ in Eq. (4), and thus we need an additional assumption that other $\mathbf{R}_j$'s are fixed when computing $\mathbf{R}_i$. Therefore, there are two options for updating $\mathbf{R}_i$: a Jacobi style update and a Gauss–Seidel style update, were the Jacobi style allows parallel updates and the Gauss–Seidel style generally obtains better convergence. We use a small number of particles so parallel updates are not greatly advantageous. Parallelism in the deformable body level gives better performance than that in the particle level. Hence, we use the Gauss–Seidel style for updating $\mathbf{R}_i$.

According to [28], the optimal rotation $\mathbf{R}$ that minimizes $\sum v_j\|\mathbf{R}\bar{\mathbf{u}}_j - \mathbf{u}_j\|^2$ can be obtained from the polar decomposition of the covariance matrix $\mathbf{C} = \sum v_j\mathbf{u}_j\bar{\mathbf{u}}_j^{\mathrm{T}}$. The polar decomposition can be computed robustly by using the singular value decomposition $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$, i.e., $\mathbf{C} = (\mathbf{U}\mathbf{V}^{\mathrm{T}})(\mathbf{V}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}) = \mathbf{R}\mathbf{S}$. In addition, we need a method to obtain the optimal rotation $\mathbf{R}$ that minimizes $\sum\|\mathbf{R}\bar{\mathbf{M}}_j - \mathbf{M}_j\|_F^2$, which can be derived by employing the definition of the Frobenius norm. Suppose that $\bar{\mathbf{M}} = [\bar{\mathbf{m}}_1|\bar{\mathbf{m}}_2|\bar{\mathbf{m}}_3]$ and $\mathbf{M} = [\mathbf{m}_1|\mathbf{m}_2|\mathbf{m}_3]$. Then, $\|\mathbf{R}\bar{\mathbf{M}}-\mathbf{M}\|_F^2 = \|\mathbf{R}[\bar{\mathbf{m}}_1|\bar{\mathbf{m}}_2|\bar{\mathbf{m}}_3]-[\mathbf{m}_1|\mathbf{m}_2|\mathbf{m}_3]\|_F^2 = \sum\|\mathbf{R}\bar{\mathbf{m}}_i-\mathbf{m}_i\|^2$. Thus, the optimal rotation $\mathbf{R}$ that minimizes $\|\mathbf{R}\bar{\mathbf{M}} - \mathbf{M}\|_F^2$ can be obtained from the polar decomposition of the covariance matrix $\mathbf{C} = \sum\mathbf{m}_i\bar{\mathbf{m}}_i^{\mathrm{T}}$, which can be further written as $\mathbf{C} = [\mathbf{m}_1|\mathbf{m}_2|\mathbf{m}_3][\bar{\mathbf{m}}_1|\bar{\mathbf{m}}_2|\bar{\mathbf{m}}_3]^{\mathrm{T}} = \mathbf{M}\bar{\mathbf{M}}^{\mathrm{T}}$. Therefore, the optimal rotation $\mathbf{R}$ that minimizes $\sum\|\mathbf{R}\bar{\mathbf{M}}_j - \mathbf{M}_j\|_F^2$ can be obtained from the polar decomposition of $\mathbf{C} = \sum\mathbf{M}_j\bar{\mathbf{M}}_j^{\mathrm{T}}$.

Then, the optimal rotation $\mathbf{R}_i$ that minimizes the energy given in Eq. (4) can be obtained from the polar decomposition of the following matrix:

$$\mathbf{C}_i^e = k\sum_{j\in\mathcal{N}_i}\Big[\mathbf{R}_j(\bar{\mathbf{A}}_j + \bar{\mathbf{A}}_i) + v_j(\mathbf{x}_j - \mathbf{x}_i)(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)^{\mathrm{T}}\Big], \tag{5}$$

where $\bar{\mathbf{A}}_j = \bar{\mathbf{E}}_j\mathbf{A}_j\bar{\mathbf{E}}_j^{\mathrm{T}}$ is the moment matrix of the ellipsoid $\mathcal{E}_j$ oriented with $\bar{\mathbf{E}}_j$. The orientation of the ellipsoid $\mathcal{E}_i$ is updated with this optimal rotation: $\mathbf{E}_i = \mathbf{R}_i\bar{\mathbf{E}}_i$.

Optimal position. To compute the optimal positions from the given rotations, we compute the gradient of $E^e(\mathcal{G})$ with respect to the positions $\mathbf{x}_i$'s and set them to zero: we seek the solution of $\partial E^e(\mathcal{G})/\partial\mathbf{x}_i = 0$. By differentiating Eq. (4) with respect to $\mathbf{x}_i$, we can obtain a set of partial derivatives:

$$\frac{\partial E^e(\mathcal{G})}{\partial\mathbf{x}_i} = k\sum_{j\in\mathcal{N}_i}\Big[(v_j + v_i)(\mathbf{x}_i - \mathbf{x}_j) - (v_j\mathbf{R}_i + v_i\mathbf{R}_j)(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j)\Big]. \tag{6}$$

If we let $\partial E^e(\mathcal{G})/\partial\mathbf{x}_i = 0$, we obtain a sparse linear system comprising the following equations:

$$k\sum_{j\in\mathcal{N}_i}(v_j + v_i)(\mathbf{x}_i - \mathbf{x}_j) = k\sum_{j\in\mathcal{N}_i}\Big(v_j\mathbf{R}_i + v_i\mathbf{R}_j\Big)(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j). \tag{7}$$

By introducing a $3n$-dimensional vector $\mathbf{x}$ comprising $\mathbf{x}_i$, Eq. (7) can be further written compactly as

$$\mathbf{L}\mathbf{x} = \mathbf{b}, \tag{8}$$

where $\mathbf{L}$ is a $3n\times 3n$ sparse symmetric matrix and $\mathbf{b}$ is a $3n$-dimensional vector. After including the momentum potential energy or the direct manipulation constraint energy, the system matrix $\mathbf{L}$ becomes a sparse symmetric positive-definite matrix. In addition, $\mathbf{L}$ depends only on the rest state. Hence, the linear system can be solved efficiently using a precomputed sparse Cholesky factorization of $\mathbf{L}$.

3.3. Momentum potential energy

Motivated by projective dynamics, we employ a variational formulation for elastodynamic deformation. In particular, we apply the momentum potential energy employed in [4, 20] to every material point of the ellipsoidal particles. Then, the momentum potential energy of a deformation graph is defined as follows:

$$E^k(\mathcal{G}) = \frac{1}{2h^2}\sum_i\int_{\mathcal{E}_i}\rho\Big\|\mathbf{x} - 2\mathbf{x}^c + \mathbf{x}^p\Big\|^2 d\mathbf{x}, \tag{9}$$

where ρ is the density of the particle, and $\mathbf{x}^c$ and $\mathbf{x}^p$ are the current and previous positions of the material point, respectively. We note that $\mathbf{x}^c$ and $\mathbf{x}^p$ are also functions of $\mathbf{x}$.

Eq. (9) can be analytically integrated in the same manner as the elastic potential energy by using the integral properties of an ellipsoid described in Section 3.2, which yields a discrete version of the momentum potential energy:

$$E^k(\mathcal{G}) = \frac{1}{2h^2}\sum_i\Big(\Big\|[\mathbf{R}_i - 2\mathbf{R}_i^c + \mathbf{R}_i^p]\bar{\mathbf{E}}_i\mathbf{A}_i^{\frac{1}{2}}\Big\|_F^2 + m_i\Big\|\mathbf{x}_i - 2\mathbf{x}_i^c + \mathbf{x}_i^p\Big\|^2\Big), \tag{10}$$

where $m_i = \rho v_i$ is the mass of the particle $\mathcal{E}_i$, and $\mathbf{R}_i^c$, $\mathbf{R}_i^p$, $\mathbf{x}_i^c$ and $\mathbf{x}_i^p$ are its current and previous rotations and positions, respectively. Note that the rotational terms finally appear explicitly in the discrete version.

Optimal rotation. In a similar manner to deriving Eq. (5) from Eq. (4), the optimal rotation $\mathbf{R}_i$ that minimizes the energy given in Eq. (10) can be obtained from the polar decomposition of the following matrix:

$$\mathbf{C}_i^k = \frac{1}{h^2}\Big(2\mathbf{R}_i^c - \mathbf{R}_i^p\Big)\bar{\mathbf{A}}_i. \tag{11}$$

Hence, $\mathbf{R}_i$ that minimizes $E^k(\mathcal{G}) + E^e(\mathcal{G})$ can be obtained from the polar decomposition of $\mathbf{C}_i^k + \mathbf{C}_i^e$.

Table 1. Model statistics and performance data.

Example	Fig.	Deform. Graph Nodes	Deform. Graph Edges	Surface Vertices	Surface Faces	Solver n_s	Solver n_i	Computation time per frame* (ms) Local	Global	Collsn.	Total	FPS Maya	FPS App.
Bar manipulation	1	20	19	674	672	1	10	0.27	0.07	·	0.34	60.0	·
Bar stiffness	2	20	19	674	672	1	10	0.27	0.07	·	0.34	60.0	·
Dinosaur inverted	4	100	261	4,334	8,664	1	10	0.70	0.24	·	0.94	60.0	·
Dinosaur falling	5	100	261	4,334	8,664	3	3	0.74	0.24	0.16	1.15	60.0	·
Sheet	6	256	480	400	1440	1	10	1.63	0.58	0.09	2.32	60.0	·
Bunny and sheet	6	496	1,647	3,251	5,344	6	2	5.60	2.15	4.71	12.48	36.5	·
52 models (4 CPU cores)	7	4,120	18,574	175K	350K	6	2	7.86	2.91	15.46	26.24	·	36.32
180 models (4 CPU cores)	7	4,404	18,640	704K	1,406K	6	2	8.87	3.28	14.97	27.14	·	30.13

n_s and n_i are the numbers of sub-steps and iterations, respectively. *The per iteration time can be obtained by dividing by $(n_s \times n_i)$.

Optimal position. In addition, the partial derivatives required for the optimal positions can be computed as follows:

$$\frac{\partial E^k(\mathcal{G})}{\partial \mathbf{x}_i} = \frac{m_i}{h^2}\left(\mathbf{x}_i - 2\mathbf{x}_i^c + \mathbf{x}_i^p\right). \tag{12}$$

Then, the linear system given in Eq. (7) needs to be rewritten after adding each side of Eqs. (6) and (12). Consequently, the system matrix $\mathbf{L}$ becomes positive-definite.

3.4. Constraints

Position constraint. We support direct manipulation of the deformable body through a position constraint, which transforms the oriented particles so a visual vertex $\bar{\mathbf{v}}_c$ at the rest state is located at $\mathbf{v}_c$. Thus, we define an energy to measure the squared distance between the position obtained using the linear blend skinning of $\bar{\mathbf{v}}_c$ and the user-specified position $\mathbf{v}_c$, as follows:

$$E_p^c(\mathcal{G}) = \frac{w_p^c}{2}\left\| \sum_{j\in\mathcal{K}_c} w_j(\bar{\mathbf{v}}_c)[\mathbf{R}_j(\bar{\mathbf{v}}_c - \bar{\mathbf{x}}_j) + \mathbf{x}_j] - \mathbf{v}_c \right\|^2, \tag{13}$$

where w_p^c is a weight for the total energy $E(\mathcal{G})$ and $\mathcal{K}_c$ denotes the k-nearest particles of $\bar{\mathbf{v}}_c$.

The positional constraint energy defined above affects the i-th oriented particles directly for $i \in \mathcal{K}_c$. In order to consider this positional constraint during the computation of $\mathbf{R}_i$, we rewrite $E_p^c(\mathcal{G})$ from the perspective of $\mathbf{R}_i$ in the form: $E_p^c(\mathcal{G}) = \frac{1}{2}w_p^c\|\mathbf{R}_i\bar{\mathbf{u}}_i - \mathbf{u}_i\|^2$, where $\bar{\mathbf{u}}_i = w_i(\bar{\mathbf{v}}_c - \bar{\mathbf{x}}_i)$ and $\mathbf{u}_i = \mathbf{v}_c - \sum_{j\neq i} w_j\mathbf{R}_j(\bar{\mathbf{v}}_c - \bar{\mathbf{x}}_j) - \sum w_j\mathbf{x}_j$. Then, the covariance matrix $\mathbf{C}_i^p = w_p^c\mathbf{u}_i\bar{\mathbf{u}}_i^{\mathrm{T}}$ needs to be incorporated in the polar decomposition for $\mathbf{R}_i$. Consequently, $\mathbf{R}_i$ should be obtained from the polar decomposition of $(\mathbf{C}_i^k + \mathbf{C}_i^e + \mathbf{C}_i^p)$ when $i \in \mathcal{K}_c$.

Similarly, we need to address the additional partial derivatives of $E_p^c(\mathcal{G})$ with respect to $\mathbf{x}_i$ when computing $\mathbf{x}_i$ for $i \in \mathcal{K}_c$:

$$\frac{\partial E_p^c(\mathcal{G})}{\partial \mathbf{x}_i} = w_p^c w_i\Big(\sum_{j\in\mathcal{K}_c} w_j(\bar{\mathbf{v}}_c)[\mathbf{R}_j(\bar{\mathbf{v}}_c - \bar{\mathbf{x}}_j) + \mathbf{x}_j] - \mathbf{v}_c\Big). \tag{14}$$

Thus, the linear system given in Eq. (7) needs to be rewritten after adding each side of Eqs. (6) and (12) for all i, and (14) for $i \in \mathcal{K}_c$.

Direction constraint. We also support direct manipulation through a direction constraint, which transforms the oriented particles so a unit vector $\bar{\mathbf{n}}_c$ on a visual vertex is aligned with $\mathbf{n}_c$. We define an energy for the direction constraint as follows:

$$E_d^c(\mathcal{G}) = \frac{w_d^c}{2}\left\| \frac{\sum_{j\in\mathcal{K}_c} w_j(\bar{\mathbf{v}}_c)\mathbf{R}_j\bar{\mathbf{n}}_c}{\left\|\sum_{j\in\mathcal{K}_c} w_j(\bar{\mathbf{v}}_c)\mathbf{R}_j\bar{\mathbf{n}}_c\right\|} - \mathbf{n}_c \right\|^2, \tag{15}$$

where w_d^c is a weight for the total energy $E(\mathcal{G})$. This energy is related only to the rotations of the oriented particles. However, it is difficult to minimize $E_d^c(\mathcal{G})$ directly due to the renormalization of the direction vector. To facilitate the optimization, we treat $m = \|\sum w_j\mathbf{R}_j\bar{\mathbf{n}}_c\|$ as a constant during the local optimization step for $\mathbf{R}_i$, although it is a function of $\mathbf{R}_i$. Then, $E_d^c(\mathcal{G})$ can be written in the form: $E_d^c(\mathcal{G}) = \frac{1}{2}w_d^c\|\mathbf{R}_i\bar{\mathbf{r}}_i - \mathbf{r}_i\|^2$, where $\bar{\mathbf{r}}_i = \frac{w_i}{m}\bar{\mathbf{n}}_c$ and $\mathbf{r}_i = \mathbf{n}_c - \sum_{j\neq i}\frac{w_j}{m}\mathbf{R}_j\bar{\mathbf{n}}_c$ for $i \in \mathcal{K}_c$. Consequently, the covariance matrix $\mathbf{C}_i^d = w_d^c\mathbf{r}_i\bar{\mathbf{r}}_i^{\mathrm{T}}$ should be incorporated in the polar decomposition for $\mathbf{R}_i$, i.e., $\mathbf{R}_i$ can be obtained from the polar decomposition of $(\mathbf{C}_i^k + \mathbf{C}_i^e + \mathbf{C}_i^d)$ when $i \in \mathcal{K}_c$.

4. Experimental results

The proposed method was implemented as an Autodesk MAYA plug-in for contents creation and offline rendering. It was also implemented as an OpenGL application with GLSL shaders for real-time demonstration. All of the experiments were performed using a MacBook Pro laptop computer with an Intel Core I7 2.9 GHz CPU, 16 GB SDRAM, and ATI Radeon Pro 460 4 GB VRAM. The time step size was fixed to $h = 1/30s$ but the number of sub-steps depended on the scenario. Three sub-steps were employed for the collision with the floor and six sub-steps for the collision between deformable models. The total number of iterations for the local/global solver in a single time step was maintained at 10. The model statistics and performance data are summarized in Table 1. The results are demonstrated in the accompanying video.

The first experiment aimed to demonstrate that the proposed method can simulate one-dimensional models robustly even with a small number of particles connected serially. Fig. 1 shows an elastic bar twisted and manipulated with position/direction constraints. The right inset illustrates the underlying oriented particles. The deformation graph comprises 20 particles and 19 edges. Nevertheless, the simulation is robust and the result is interesting. All the vertices on one end of

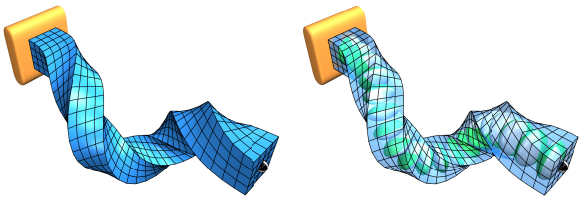

Fig. 1. Twisting and manipulating a bar with a serial deformation graph.

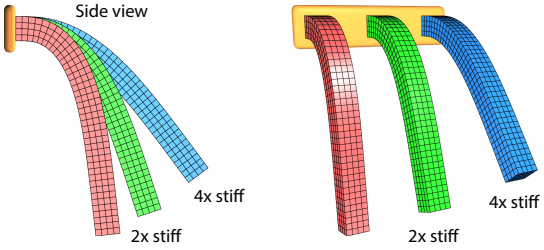

Fig. 2. Bars deformed under gravity with different stiffness values. The green and blue bars are two and four times stiffer than the red.

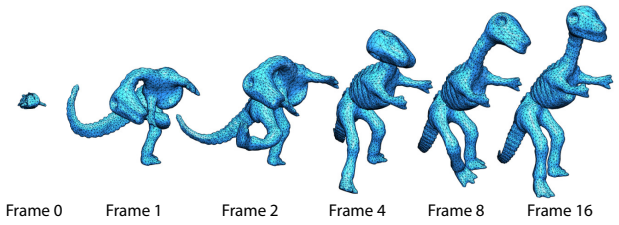

Fig. 4. A dinosaur returns to its rest pose from an extremely deformed pose.

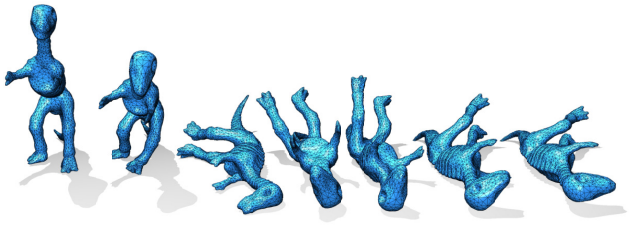

Fig. 5. A dinosaur falling down on the floor.

the bar are fixed with position constraints, and a single vertex at the center of the other end is manipulated with a single position constraint and three direction constraints.

The next experiment aimed to verify that the material stiffness is simulated appropriately with different stiffness constants k. Fig. 2 shows the steady states of three elastic bars deformed under gravity with different stiffness values. The vertices on one end of each bar were fixed as in the previous experiment. The green and blue bars were two and four times stiffer than the red one, respectively. In contrast to the original oriented particles approach [3], the material stiffness is largely independent of the iteration count, as shown in Fig. 3. We note that the original oriented particles approach requires a large iteration count to simulate a moderately stiff bar represented with serially connected particles. In addition, as demonstrated in the accompanying video, as the time step size becomes smaller, the simulation becomes more responsive in our method. Nevertheless, the stiffness is largely independent of the time step size.

Our method computes the positions and orientations of the particles directly via energy minimization, and thus it is extremely stable even in the situation depicted in Fig. 4. At the

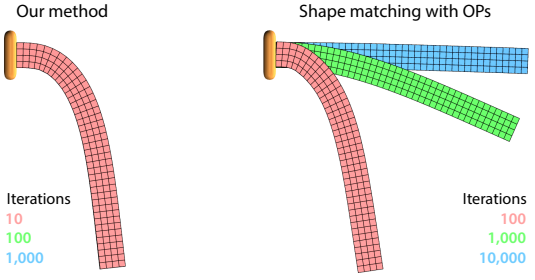

Fig. 3. The material stiffness is largely independent of the iteration count in our method (left) in contrast to shape matching [3] with oriented particles (right). The iteration counts are 10, 100, and 1000 for the left inset, and 100, 1000, and 10,000 for the right inset.

beginning of the simulation, all the particles were placed so they coincided with the center particle. We fixed the position and orientation of the center particle during the simulation. The dinosaur returned to its rest pose rapidly from an extremely deformed pose. When extreme deformation is involved, we need to take a special care during the polar decomposition of the local optimization step. The set of particles connected directly to a particle can be inverted so the determinant of the covariance matrix given in Eq. (5) may be negative, as in the case for the deformation gradient in the inverted finite element [29]. Shape matching [28] using the singular value decomposition employed in our method deals with this problem robustly by rectifying a negative eigenvalue.

In the next experiment, we let the dinosaur fall down on the floor, as shown in Fig. 5. Collisions between the ellipsoidal particles and the floor were detected as described by Müller and Chentanez [3]. The ellipsoids interpenetrating the floor were handled by pushing them back so they touched the floor. Collisions could be missed by discrete collision detection when the particles moved fast. Thus, we divided each time step into three sub-steps and we handled the collisions and frictions at the end of each sub-step. Friction was handled simply by changing the velocities of the colliding particles. The linear velocity $\mathbf{v}_i^c = (\mathbf{x}_i^c - \mathbf{x}_i^p)/h$ of the i-th particle can be employed as a state variable for the position with $\mathbf{x}_i^c$, so $(2\mathbf{x}_i^c - \mathbf{x}_i^p)$ becomes $(\mathbf{x}_i^c + h\mathbf{v}_i^c)$ in Eqs. (10) and (12). In a similar manner, $\dot{\mathbf{R}}_i^c = \omega_i^c \times \mathbf{R}_i^c = (\mathbf{R}_i^c - \mathbf{R}_i^p)/h$ and $\mathbf{R}_i^c$ can be employed for rotation, so $(2\mathbf{R}_i^c - \mathbf{R}_i^p)$ becomes $(\mathbf{R}_i^c + h\dot{\mathbf{R}}_i^c)$ in Eqs. (10) and (11).

Our method can simulate two-dimensional models, as shown in Fig. 6, where in the left inset, two position constraints are used to attach a deformable sheet to a rigid rod. The motion of the rod was scripted and the sheet was simulated accordingly in order to produce dynamic deformation. In the right inset, a bunny falls down on a sheet, where the four corners of the sheet are constrained with position constraints. The bunny and sheet are both deformable. Hence, collision handling was required among the ellipsoidal particles. We employed broad-

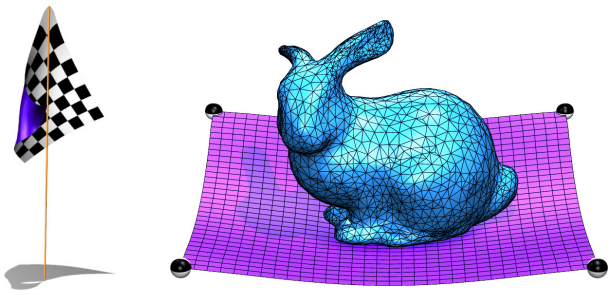

Fig. 6. Two-dimensional structures. A deformable sheet is attached to a rotating rod (left) and a deformable bunny falls down on a deformable sheet (right).

phase culling based on spatial hashing [30], and the efficient narrow-phase collision detection and response method [3]. A pair of colliding ellipsoids was resolved by separating them outward in the inter-center direction.

For real-time simulation and rendering of many deformable bodies, we exploited a multi-core CPU and many-core GPU. Fig. 7 shows a screenshot of an OpenGL application developed with GLSL shaders. The simulation was performed with a simple parallelization using OpenMP and GPU skinning. The scene in the bottom left inset comprises 52 deformable models with 4.1 K particles and 18.6 K edges for the deformation graphs, and 175 K vertices and 350 K triangles for the visual surfaces. As the simulation progressed, 40 deformable models (10 instances each of the dinosaur, bunny, armadillo, and torus) fell down on the floor fenced by 12 deformable columns. The simulation ran at over 30 FPS with six sub-steps for collision detection and response. The number of sub-steps was large in this experiment, and thus only two iterations of the local/global solver were used. On average, local optimization, global optimization, and collision handling required 25, 13, and 62 percent of the total computational time, respectively.

The final experiment demonstrated the effectiveness of our technique as an embedded space deformation technique. Thus, for the example shown in the bottom right inset of Fig. 7, we employed a smaller number of particles for each deformable model (illustrated in the top right inset) than the previous example in order to simulate a larger number of deformable models. In this example, each dinosaur, bunny, and armadillo comprised 30 particles, and each torus comprised 12 particles. In the previous examples, each model comprised 100 particles. Nevertheless, the simulation was robust and responsive. The simulation ran at over 30 FPS with 180 deformable models comprising 4.4 K particles and 18.6 K edges for the deformation graphs, and 704 K vertices and 1,406 K triangles for the visual surfaces.

5. Conclusion

In this paper, we proposed an ARAP approach for the real-time simulation of physics-based deformation with oriented particles. The proposed method can deal with one-, two-, and three-dimensional deformable models robustly even with a small number of particles by extending the deformation graph [27] with oriented particles [3] and by formulating the corresponding ARAP deformation energy. Our method employs an implicit Euler integration scheme formulated as an energy minimization problem [4] where it seeks the optimal positions and orientations of the particles using an alternating local/global optimization solver, which is easy to implement and computationally efficient. The material stiffness is largely independent of the iteration count and the time step size, in contrast to shape matching with oriented particles [3]. Our experimental results demonstrate that hundreds of deformable models with an extremely large number of polygons can be simulated robustly in real time using thousands of particles.

Minimizing the ARAP deformation energy is analogous to enforcing the shape matching constraint for the local rigidity. Our method is not intended for physically correct simulations but instead it is a practical approach for obtaining visually plausible real-time deformations. Our formulation utilizes integration over ellipsoids but the ellipsoids are discrete approximations of a deformable body. Hence, the physical behavior depends on the discretization, which is an inherent limitation of approaches based on oriented particles. Our formulation also inherits the slow convergence problem from projective dynamics, although this can be circumvented to a certain extent by using a Chebyshev semi-iterative approach [23] or a quasi-Newton method [25]. Another shortcoming is that the global optimization solver exploits pre-factorization of the system matrix, which must be re-factored to deal with run-time tearing or fracturing. Fortunately, we developed our formulation by considering a moderate number of particles. Building and factoring 300 particles requires less than 2 ms. Hence, we plan to incorporate plasticity and fracturing as described by Choi [31]. Currently, collision detection is the performance bottleneck in a large scene because we employ a naive parallelization of spatial hashing. We plan to exploit more elaborate parallelizations in the CPU/GPU. The ARAP deformation energy only supports isotropic materials, but we aim to consider anisotropic materials by introducing additional deformation energies in future research.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (NRF-2014R1A1A1008486, NRF-2017R1D1A1B03035685) and it was conducted during the sabbatical year of Kwangwoon University in 2015.

References

[1] Müller, M, Heidelberger, B, Hennix, M, Ratcliff, J. Position based dynamics. In: Proc. Virtual Reality Interactions and Physical Simulations. 2006, p. 71–80.

[2] Müller, M, Heidelberger, B, Teschner, M, Gross, M. Meshless deformations based on shape matching. ACM Transactions on Graphics (Proc ACM SIGGRAPH 2005) 2005;24(3):471–478.

[3] Müller, M, Chentanez, N. Solid simulation with oriented particles. ACM Transactions on Graphics (Proc ACM SIGGRAPH 2011) 2011;30(4):92:1–92:9.

[4] Bouaziz, S, Martin, S, Liu, T, Kavan, L, Pauly, M. Projective dynamics: Fusing constraint projections for fast simulation. ACM Transactions on Graphics 2014;33(4):154:1–154:11.

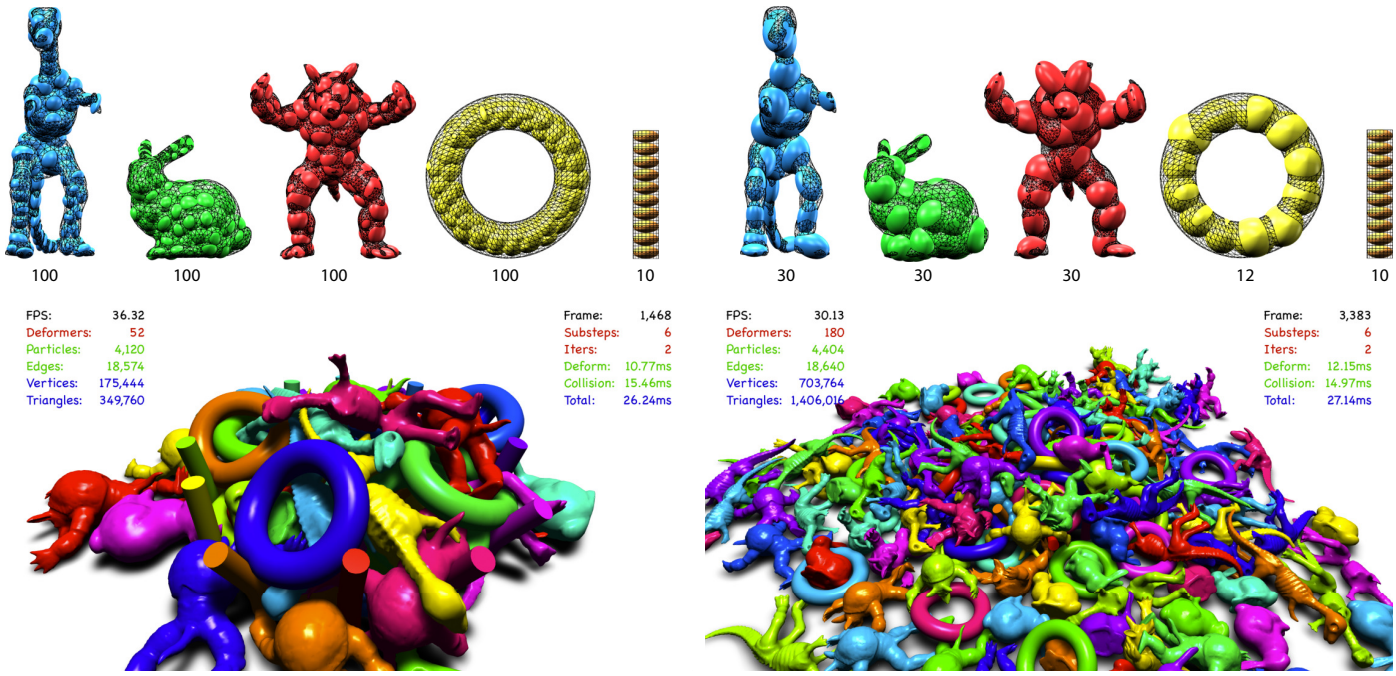

Fig. 7. Real-time simulations of 52 deformable models (bottom left) with a relatively large number of particles per model (top left) and 180 models (bottom right) with a small number of particles per model (top right).

[5] Terzopoulos, D, Fleischer, K. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. Computer Graphics (Proc ACM SIGGRAPH 88) 1988;22(4):269–278.

[6] Nealen, A, Müller, M, Keiser, R, Boxerman, E, Carlson, M. Physically based deformable models in computer graphics. Computer Graphics Forum 2006;25(4):809–836.

[7] Rivers, AR, James, DL. FastLSM: Fast lattice shape matching for robust real-time deformation. ACM Transactions on Graphics (Proc ACM SIGGRAPH 2007) 2007;26(3):82:1–82:6.

[8] Rungjiratananon, W, Kanamori, Y, Nishita, T. Chain shape matching for simulating complex hairstyles. Computer Graphics Forum 2010;29(8):2438–2446.

[9] Diziol, R, Bender, J, Bayer, D. Robust real-time deformation of incompressible surface meshes. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2011, p. 237–246.

[10] Bender, J, Weber, D, Diziol, R. Fast and stable cloth simulation based on multi-resolution shape matching. Computers & Graphics 2013;37(8):945–954.

[11] Steinemann, D, Otaduy, MA, Gross, M. Fast adaptive shape matching deformations. In: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2008, p. 87–94.

[12] Macklin, M, Müller, M. Position based fluids. ACM Transactions on Graphics 2013;32(4):104:1–104:6.

[13] Deul, C, Charrier, P, Bender, J. Position-based rigid body dynamics. Computer Animation and Virtual Worlds 2014;27(2):103–112.

[14] Umetani, N, Schmidt, R, Stam, J. Position-based elastic rods. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2014,.

[15] Kugelstadt, T, Schoemer, E. Position and orientation based cosserat rods. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2016,.

[16] Müller, M, Chentanez, N, Kim, TY, Macklin, M. Strain based dynamics. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2014, p. 149–157.

[17] Macklin, M, Müller, M, Chentanez, N, Kim, TY. Unified particle physics for real-time applications. ACM Transactions on Graphics 2014;33(4):153:1–153:12.

[18] Bender, J, Müller, M, Otaduy, MA, Teschner, M, Macklin, M. A survey on position-based simulation methods in computer graphics. Computer Graphics Forum 2014;33(6):228–251.

[19] Macklin, M, Müller, M, Chentanez, N. XPBD: Position-based simulation of compliant constrained dynamics. In: Proc. ACM Motion in Games. 2016, p. 49–54.

[20] Liu, T, Bargteil, AW, O'Brien, JF, Kavan, L. Fast simulation of mass-spring systems. ACM Transactions on Graphics 2013;32(6):214:1–214:7.

[21] Kharevych, L, Yang, W, Tong, Y, Kanso, E, Marsden, JE, Schröder, P, et al. Geometric, variational integrators for computer animation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2006, p. 43–51.

[22] Gast, TF, Schroeder, C, Stomakhin, A, Jiang, C, Teran, JM. Optimization integrator for large time steps. IEEE Transactions on Visualization and Computer Graphics 2015;21(10):1103–1115.

[23] Wang, H. A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. ACM Transactions on Graphics 2015;34(6):246:1–246:9.

[24] Narain, R, Overby, M, Brown, GE. ADMM ⊇ projective dynamics: Fast simulation of general constitutive models. In: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2016, p. 21–28.

[25] Liu, T, Bouaziz, S, Kavan, L. Quasi-Newton methods for real-time simulation of hyperelastic materials. ACM Transactions on Graphics 2017;36(3):23:1–23:16.

[26] Sorkine, O, Alexa, M. As-rigid-as-possible surface modeling. In: Proc. Eurographics Symposium on Geometry Processing. 2007, p. 109–116.

[27] Sumner, RW, Schmid, J, Pauly, M. Embedded deformation for shape manipulation. ACM Transactions on Graphics 2007;26(3):80:1–80:8.

[28] Umeyama, S. Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 1991;13(4):376–380.

[29] Irving, G, Teran, J, Fedkiw, R. Invertible finite elements for robust simulation of large deformation. In: Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2004, p. 131–140.

[30] Teschner, M, Heidelberger, B, Müller, M, Pomeranets, D, Gross, M. Optimized spatial hashing for collision detection of deformable objects. In: Proc. Vision, Modeling, Visualization. 2003, p. 47–54.

[31] Choi, MG. Real-time simulation of ductile fracture with oriented particles. Computer Animation and Virtual Worlds (Proc CASA 2014) 2014;25(3-4):455–463.