Noname manuscript No.
(will be inserted by the editor)

Meshless Methods for Physics-Based Modeling and Simulation of Deformable Models

Xiaohu Guo · Hong Qin

Received: date / Accepted: date

Abstract As 3D digital photographic and scanning devices produce higher resolution images, acquired geometric data sets grow more complex in terms of the modeled objects' size, geometry, and topology. As a consequence, point-sampled geometry is becoming ubiquitous in graphics and geometric information processing, and poses new challenges which have not been fully resolved by the state-of-art graphical techniques. In this paper, we address the challenges by proposing a meshless computational framework for dynamic modeling and simulation of solids and thin-shells represented as point samples. Our meshless framework can directly compute the elastic deformation and fracture propagation for any scanned point geometry, without the need of converting them to polygonal meshes or higher order spline representations. We address the necessary computational techniques, such as Moving Least Squares, Hierarchical Discretization, and Modal Warping, to effectively and efficiently compute the physical simulation in real-time. This meshless computational framework aims to bridge the gap between the point-sampled geometry with physics-based modeling and simulation governed by partial differential equations.

Keywords Meshless Method · Physics-Based Modeling · Physics-Based Simulation · Deformable Models

Xiaohu Guo is partially supported by National Science Foundation CCF-0727098. Hong Qin is partially supported by National Science Foundation IIS-0710819.

X. Guo
Department of Computer Science
University of Texas at Dallas
Tel: +1-972-883-4723
Fax: +1-972-883-2349
E-mail: xguo@utdallas.edu

H. Qin
Department of Computer Science
Stony Brook University
Tel: +1-631-632-8450
Fax: +1-631-632-8334
E-mail: qin@cs.sunysb.edu

1 Introduction

A fundamental question in digital modeling and simulation is the choice of geometric and functional representation: What mathematical description should be chosen to represent the surface or volume of a 3D object on a digital computer? The diversity of the application fields of 3D geometry is reflected in a wide variety of geometric representations that have been proposed in the past. For example, industrial design applications for car and airplane construction are mostly based on B-splines or NURBS [9]; medical applications make frequent use of implicit representations such as level sets [19]; game and movie industry has focused on polygonal representations such as triangle meshes [7]. The choice of geometric representation is typically guided by the following considerations: the number of individual primitives, the descriptive power or approximation order, the power of handling complicated geometry, and the ease of performing topological changes.

– B-splines and NURBS can represent complex shapes with relatively few primitives, since each individual spline patch has a high approximation order. However, combining different patches to a consistent surface model of arbitrary topology can be difficult, since strict global continuity constraints need to be observed. Furthermore, it is difficult to model high-frequency features or even discontinuity. Some attractive properties such as local adaptivity and multi-resolution are rather difficult to achieve.
– Polygonal meshes are easier to handle as they are defined by a set of vertices with a consistent adjacency graph, but require a higher number of primitives to achieve the same geometric accuracy. With increasing resolution of the scanning devices, geometric data sets are becoming more and more complex, lately reaching billions of sample points for a single model [17]. Since scanners typically produce a collection of point samples, many reconstruction methods are used to convert the point cloud into a polygonal representation. Unfortunately, most of these methods do not scale well with model size, e.g., many techniques based on the Delaunay triangulation have a worst-case complexity of $O(n^2)$, where n is the number of sample points. Being able to directly process, animate, or simulate point cloud representations would avoid the need for surface reconstruction entirely.

Nowadays three-dimensional acquisition is an increasingly popular means of creating surface models. With the dramatic increase of the polygonal complexity of the acquired graphical models, large scaled point-sampled geometry is becoming ubiquitous in graphics and geometric information processing pipeline. Points have several unique advantages over traditional primitives such as triangle meshes in modeling, animation, and simulation. For example, they are free of connectivity concerns, which makes them very suitable for dynamic shape manipulation and physical simulation when the underlying geometry is undergoing topological changes.

Traditionally in computer animation and physical simulation, meshes are inevitable for physical simulations since the finite element methods (FEM) require an explicit mesh structure. However, for some complex physical effects, such as large deformation and fracture, it will pose grand technical challenges in terms of maintaining the topological consistency of the underlying meshes. In these cases, efficient and consistent surface (and volumetric) representations are necessary to facilitate geometric and topological operations.

- The finite element interpolation functions are then built upon the mesh, which ensures the compatibility of the interpolation. However, this procedure is not always advantageous, because the numerical compatibility condition is not the same as the physical compatibility condition of a continuum. For instance, in a Lagrangian type of computations, one may experience mesh distortion, which can either end the computation altogether or result in drastic deterioration of accuracy. In computer simulations of very large deformation, a distorted mesh introduces severe errors in numerical computations. Therefore, it would be computationally efficacious to discretize a continuum by only a set of nodal points without mesh constraints. This is the *leitmotif* of contemporary meshless methods in mechanical engineering, scientific computing, and computer animation.
- In simulations of failure processes [23], we need to model the propagation of cracks along arbitrary and complex paths. This problem, in particular, becomes a notoriously difficult task using conventional mesh-based computational techniques such as finite element method or finite difference method. In essence, the underlying structure of these methods, which stem from their reliance on meshes, impedes the flexible modeling and natural handling of discontinuities that do not coincide with the original mesh lines. Therefore, the most viable strategy for dealing with moving discontinuities in these methods is to remesh in each time step of the integration so that mesh lines remain coincident with the discontinuities throughout the simulation. However, this can introduce numerous difficulties for data management, such as the strong need to map between meshes in consecutive stages of the simulation, which inevitably results in degradation of both accuracy and complexity for system implementation. In addition, model remeshing becomes an unavoidable burden. In sharp contrast, meshless methods can overcome the above difficulties associated with mesh structure, and avoid complex remeshing operations and the associated problems of element cutting and mesh alignment sensitivity common in FEM.

In this paper, we try to build high-fidelity meshless computational models of the point-sampled scanned geometry in the physical world. The objective is to unify complex point-sampled geometry, easy topological change, realistic physical properties and behavior in a single meshless framework. Unlike meshes or splines, the point-sampled geometry representation has no intrinsic differential structure, which make it very hard to incorporate rigorous physical model from continuum mechanics. In this paper, we introduce our meshless simulation framework, in which the partial differential equations (for dynamic physical simulation) can be applied and solved directly over point samples via Moving Least Squares (MLS) shape functions defined on the parametric domain without explicit connectivity information. Our meshless framework can perform efficient simulation for elastic thin-shell and solid deformations. For thin-shell simulation, the simulation domain is not naturally given by the point geometry, which make it necessary to utilize parameterizations for the point-sampled surfaces. For solid deformation, the number of volumetric nodes is typically prohibitive for real-time simulation. We exploit the methodology of Modal Analysis and adapt the Modal Warping technique into our volumetric meshless simulation framework to achieve real-time manipulation and deformation.

The remainder of this paper is organized as follows. In section 2 we briefly review the prior work utilizing meshless methods for physics-based modeling and simulation. In section 3 we describe the detailed theoretical derivations for the meshless method, particularly the Moving Least Squares method. In section 4 we describe the hierarchical discretization methods for our meshless framework. For solid simulation, we use octree discretization based on implicit functions of the point-set surface (section 4.1). For thin-shell simulation, we use quadtree discretization based on the global conformal parameterization of the surfaces (section 4.2). In section 5 we describe the differential equations used to represent the solid and thin-shell dynamics, and the Modal Warping technique that we use to solve the dynamic system in real-time. Section 6 shows the statistics and performance data of our experimental results. Finally, we conclude this paper with several possible directions for future work in section 7.

2 Previous Work

Meshless (mesh-free) methods [4,18] have been developed in the field of mechanical engineering to enable solving partial differential equations (PDEs) numerically. The meshless computation is based on a set of scattered nodes without having to recourse to an additional mesh structure. The meshless methods require only a set of nodes distributed across the entire analysis domain. The shape function associated with each node is then constructed to approximate (or interpolate) the field functions using their values at the sampling nodes in the analysis domain without explicit connectivity, while satisfying certain basic requirements, such as compact support for computational accuracy and efficiency, stability and consistency to ensure numerical convergence, etc.

Meshless method was introduced into the graphics and animation field by Desbrun and Cani [8] using a particle system coated with a smooth iso-surface for animating soft inelastic substance which undergo topological changes. Later they applied Smoothed Particle Hydrodynamics (SPH) to simulate highly deformable bodies. Müller et al. [21] presented a method for modeling and animating elastic, plastic, and melting volumetric objects based on the MLS approximation of the displacement field. Most recently, they presented a geometrically motivated approach in [22] for simulating deformable point-based objects. Pauly et al. [27] simulated volumetric meshless fracture with a highly dynamic surface and volume sampling method that affords complex fracture patterns of interacting and branching cracks. Guo and Qin [11] combined meshless method with modal analysis framework to provide real-time deformation of volumetric objects. The most relevant work to our meshless thin-shell simulation is the approach proposed by Wicke et al. [29], which uses locally defined fibers to approximate the differential surface operators of the thin shell functional.

3 Meshless Methods

During the last two decades, meshless methods have been developed that enable solving PDEs numerically, based on a set of scattered nodes without having recourse to an additional mesh structure (which must be put in place for the traditional finite element methods). The unique advantages of meshless methods are multifold: (1) there is no need to generate a mesh of nodes – they only need to be scattered within the analysis domain, which is much easier to handle in principle; (2) moving discontinuities such as cracks can be naturally facilitated, since no new mesh needs to be constructed as in finite element methods, and the computational cost of remeshing at each time step can be avoided entirely; (3) properties such as spatial adaptivity (node addition or elimination) and shape function polynomial order adaptivity (approximation/interpolation types) can streamline the adaptive model refinement and simulation in both time and space; and (4) data management overhead can be minimized during simulation. In 1994, Belytschko et al. proposed the Element Free Galerkin (EFG) method [2] to solve linear elastic problems, specifically the fracture and crack growth problems [3], in which the Moving Least Squares (MLS) interpolant was employed in a Galerkin procedure. In this paper we focus on the Element Free Galerkin method, mainly because it has been well-developed with mature techniques, and it has shown a superior rate of convergence and high efficiency in modeling moving interfaces. Other variants of available mesh-free methods can also be adopted into our prototype system in a straightforward way without theoretical obstacles.

In a nutshell, the meshless methods require only a set of nodes distributed across the entire analysis domain. The shape function associated with each node is then constructed to approximate (or interpolate) the field functions using their values at the sampling nodes in the analysis domain. For finite element methods, however, the shape functions are constructed utilizing the mesh of the elements. In sharp contrast, the shape functions in meshless methods are constructed using only the sampling nodes without any connectivity, while satisfying certain basic requirements, such as compact support for computational accuracy and efficiency, stability and consistency to ensure numerical convergence, etc. The shape functions in the EFG method are constructed by using the MLS technique, or alternatively on the basis of reproducibility conditions (note that both approaches can arrive at the same expressions for the shape functions), and it can provide continuous and smooth field approximation throughout the analysis domain with any desirable order of consistency. To make our paper self-contained, we present a brief introduction of the MLS approximation for the definition of shape functions in the following subsection.

3.1 Moving Least Squares Shape Functions

The moving least squares method can be traced back to its original application in scattered data fitting, where it has been studied under different names (e.g., local regression, "LOESS", weighted least squares, etc.) [16].

We associate each node I with a positive weight function w_I of compact support. The support of the weight function w_I defines the domain of influence of the node: $\Omega_I = \{\mathbf{x} : w_I(\mathbf{x}) = w(\mathbf{x}, \mathbf{x}_I) > 0\}$, where $w(\mathbf{x}, \mathbf{x}_I)$ is the weight function associated with node I evaluated at position $\mathbf{x}$. The approximation of the field function f at a position $\mathbf{x}$ is only affected by those nodes whose weights are non-zero at $\mathbf{x}$. We denote the set of such nodes the *active set* $\mathcal{A}(\mathbf{x})$. Figure 1 illustrates the meshless computational model with rectangular and circular local influence domains, respectively.

Let $f(\mathbf{x})$ be the field function defined in the analysis domain Ω, and $f^h(\mathbf{x})$ the approximation of $f(\mathbf{x})$ at position $\mathbf{x}$. In the MLS approximation, we let

$$f^h(\mathbf{x}) = \sum_{i=1}^{m} p_i(\mathbf{x}) a_i(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}(\mathbf{x}), \tag{1}$$

where $p_i(\mathbf{x})$ are polynomial basis functions, m is the number of basis functions in the column vector $\mathbf{p}(\mathbf{x})$, and $a_i(\mathbf{x})$ are their coefficients, which are functions of the spatial coordinates $\mathbf{x}$. In our implementation, we utilize 3-D linear basis functions: $\mathbf{p}^T_{(m=4)} =$

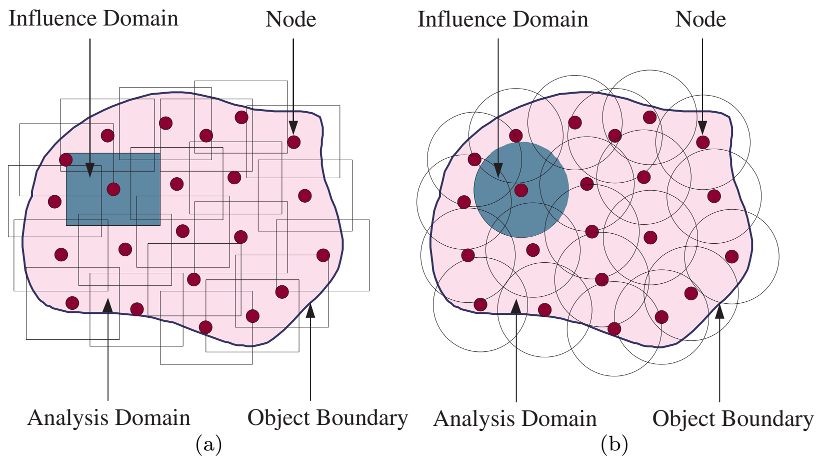

Fig. 1 The meshless computational model with rectangular (a) and circular (b) support, respectively.

$\{1, x, y, z\}$ in the interest of time performance. We can derive $\mathbf{a}(\mathbf{x})$ by minimizing a weighted L_2 norm:

$$J = \sum_{I \in \mathcal{A}(\mathbf{x})} w(\mathbf{x} - \mathbf{x}_I)[\mathbf{p}^T(\mathbf{x}_I)\mathbf{a}(\mathbf{x}) - f_I]^2, \tag{2}$$

where f_I is the nodal field value associated with the node I. We can rewrite Equation (2) in the form:

$$J = (\mathbf{Pa} - \mathbf{f})^T \mathbf{W}(\mathbf{x})(\mathbf{Pa} - \mathbf{f}), \tag{3}$$

where

$$\mathbf{f}^T = (f_1, f_2, \ldots f_n),$$

$$\mathbf{P} = \begin{bmatrix} p_1(\mathbf{x}_1) & p_2(\mathbf{x}_1) & \cdots & p_m(\mathbf{x}_1) \\ p_1(\mathbf{x}_2) & p_2(\mathbf{x}_2) & \cdots & p_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\mathbf{x}_n) & p_2(\mathbf{x}_n) & \cdots & p_m(\mathbf{x}_n) \end{bmatrix},$$

and

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x} - \mathbf{x}_1) & 0 & \cdots & 0 \\ 0 & w(\mathbf{x} - \mathbf{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w(\mathbf{x} - \mathbf{x}_n) \end{bmatrix}.$$

To find the coefficients $\mathbf{a}(\mathbf{x})$, we obtain the extremum of J by setting

$$\frac{\partial J}{\partial \mathbf{a}} = \mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{f} = 0, \tag{4}$$

where the $m \times m$ matrix $\mathbf{A}$ is called *moment matrix*:

$$\mathbf{A}(\mathbf{x}) = \mathbf{P}^T \mathbf{W}(\mathbf{x})\mathbf{P},$$

$$\mathbf{B}(\mathbf{x}) = \mathbf{P}^T \mathbf{W}(\mathbf{x}).$$

So we can obtain:

$$\mathbf{a}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\mathbf{f}. \tag{5}$$

And the shape functions are given by:

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots \phi_n(\mathbf{x})] = \mathbf{p}^T(\mathbf{x})\mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x}). \tag{6}$$

If we consider the field function as a function of both space and time $f(\mathbf{x}, t)$, the approximation in the analysis domain Ω can be written as:

$$f(\mathbf{x}, t) \approx f^h(\mathbf{x}, t) = \sum_{I \in \mathcal{A}(\hat{\mathbf{x}})} \phi_I(\mathbf{x}) f_I(t), \tag{7}$$

The moment matrix $\mathbf{A}$ may be ill-conditioned when (i) the basis functions $\mathbf{p}(\mathbf{x})$ are (almost) linearly dependent, or (ii) there are not enough nodal supports overlapping at the given point, or (iii) the nodes whose supports overlap at the point are arranged in a special pattern, such as a conic section for a complete quadratic polynomial basis $\mathbf{p}(\mathbf{x})$. Note that the necessary condition for the matrix A to be invertible is

$$\forall \mathbf{x} \in \Omega \quad card\{I : \mathbf{x} \in \Omega_I\} > m, \tag{8}$$

which can be automatically guaranteed using the octree-based (for volumetric models) and quadtree-based (for surface models) node placement method to be explained in the next section. Furthermore, the spatial derivatives of the shape functions can be obtained by noting that the differentiation of Equation (4) yields:

$$\mathbf{A}_{,i}\mathbf{a} + \mathbf{A}\mathbf{a}_{,i} = \mathbf{B}_{,i}\mathbf{f},$$

where $\mathbf{a}_{,i}$ denotes $\frac{\partial \mathbf{a}}{\partial x_i}$. Then we can obtain the derivative of $\mathbf{a}_{,i}$ by:

$$\mathbf{a}_{,i} = \mathbf{A}^{-1}(\mathbf{B}_{,i}\mathbf{f} - \mathbf{A}_{,i}\mathbf{a}). \tag{9}$$

So the factorization of $\mathbf{A}$ in Equation (6) can be re-used for the computation of the derivatives with little extra cost.

3.2 Basis and Weight Functions

To obtain a certain consistency of any desirable order of approximation, it is necessary to have a complete basis. The basis functions $\mathbf{p}(\mathbf{x})$ may include some special terms such as singularity functions, in order to ensure the consistency of the approximation and to improve the accuracy of the results. The following gives two examples of complete bases in 3 dimensions for first and second order consistency:

$$Linear : \mathbf{p}^T_{(m=4)} = \{1, x, y, z\}, \tag{10}$$

$$Quadratic : \mathbf{p}^T_{(m=10)} = \{1, x, y, z, x^2, xy, xz, y^2, yz, z^2\}. \tag{11}$$

The weight functions $w(\mathbf{x}, \mathbf{x}_I)$ play important roles in constructing the shape functions. They should be positive to guarantee a unique solution for $\mathbf{a}(\mathbf{x})$; they should decrease in magnitude as the distance to the node increases to enforce local neighbor influence; they should have compact supports, which ensure sparsity of the global matrices. They can differ in both the shape of the domain of influence (e.g., parallelepiped centered at the node for tensor-product weights, or sphere), and in functional form (e.g., polynomials of varying degrees, or non-polynomials such as the truncated Gaussian weight). In our implementation, we choose the parallelepiped domain of influence for the ease of performing numerical integrations, and utilize the composite quadratic tensor-product weight function:

$$w(\mathbf{x}, \mathbf{x}_I) = c(\frac{x - x_I}{d_{mxI}})c(\frac{y - y_I}{d_{myI}})c(\frac{z - z_I}{d_{mzI}}), \tag{12}$$

where d_{mxI}, d_{myI}, and d_{mzI} denote half of the length of the supporting parallelepiped sides (for the tensor-product weights) along three directions, respectively. The component function $c(s)$ is analytically defined as:

$$c(s) = \begin{cases} (1 - 2s^2) & for \quad 0.5 > s \geq 0 \\ 2(1 - s)^2 & for \quad 1 > s \geq 0.5 \\ 0 & for \quad s \geq 1 \end{cases} \tag{13}$$

One key attractive property of MLS approximations is that their continuity is directly related to the continuity of the weighting functions. Thus, a lower-order polynomial basis $\mathbf{p}(\mathbf{x})$ such as the linear one can still be used to generate highly continuous approximations by choosing appropriate weight functions with certain smoothness requirements. Therefore, compared to the finite element method, there is no need for post-processing to generate smooth stress and strain fields. This can facilitate the direct and fast visualization of the physical properties of volumetric objects for mechanical analysis. Note that the FEM equivalents can also be reached if the weight functions are defined as piecewise-constant entities over each influence domain.

4 Hierarchical Discretization for Meshless Dynamics

The general idea of meshless methods is to create overlapping patches Ω_I comprising a cover $\{\Omega_I\}$ of the domain Ω with shape function ϕ_I subordinate to the cover Ω_I. One way to create the meshless discretization is to start from an arbitrarily distributed set of nodes. No fixed connections between the nodes are required. The nodes are the centers of the overlapping patches Ω_i, which can be either parallelepiped or spherical domains. However, due to the rather unstructured distribution of nodes over the domain some algorithmic issues may arise: (1) a discretization without structure does not allow determination of the patches that contribute to a certain integration point without performing an expensive global search; (2) the moment matrix $\mathbf{A}$ in moving least squares shape function may become invertible if the patch covering conditions (e.g., Equation 8) are not satisfied; (3) the effective handling of the interaction between scattered nodes with the geometric boundary (e.g. the surface of a volumetric model in our solid simulation system) becomes very difficult. From the implementation's point of view, it is very important that the patches are clearly defined. The interaction between the patches themselves, and between the patches and the boundary, has to be well understood and easily accessible during the runtime of the system execution. These problems can be solved perfectly with the assistance of octree discretization (for solid models) and quadtree discretization (for thin-shell models).

4.1 Octree-Based Discretization for Solid Models

In our meshless volumetric simulation system, the input data is an unstructured point cloud comprising a closed manifold surface. If we conduct our dynamic simulation solely on surface points, many difficulties arise. First, performing inside/outside tests based entirely on surface point information is a forbidding task with many ambiguities. Second, point insertion is unavoidable if a deformation is large and in fact spreads out across the model rather significantly, in which case gaps will occur at the current resolution. To ameliorate, we compute a volumetric distance field for the input surface points. Such a distance field, which expands to the entire volumetric domain, will also aid in the selection of volumetric points at the interior of solid objects for the dynamic simulation governed by the EFG method. Let us first briefly review some relevant work which leads to the construction of octree-based distance fields for point surfaces. Pauly et al. [26] rely on the moving least squares surface projection operator for both inside/outside tests and point insertion. However, ambiguities would still occur in many degenerate cases if we only use the moving least squares surface projection operator [30]. Guo et al. [10] proposed to embed the point set surfaces into volumetric scalar fields to facilitate surface representation, surface editing, dynamic point re-sampling, collision detection, etc. Implicit surfaces can be considered a natural and powerful tool for modeling unstructured point set surfaces for the following reasons: (i) the inside/outside test can be performed by directly utilizing the implicit function; (ii) the topology of the implicit surface can be easily updated without any ambiguity. Figure 2 (a) shows the visualization of distance fields using a color-coded 2D slice.

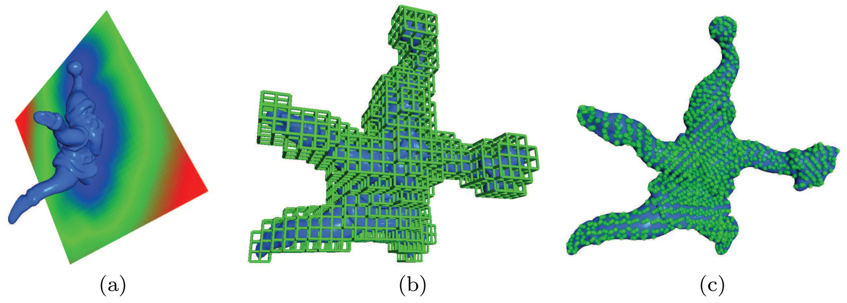

Fig. 2 (a) Distance field visualization using a color-coded 2D slice; (b) Octree-based discretization; (c) Volumetric node placement.

In our implementation, we utilize *multi-level partition of unity*(MPU) implicit surface construction method proposed by Ohtake et al. [24]. The multi-level approach allows us to construct implicit surface models from large point sets based on an octree subdivision method that adapts to variations in the complexity of the local shape. We also observed that the octree discretization of the volume can provide a structure to construct the patches which would provide a priori information with respect to the size and interactions of the patches [15]. The octree subdivides the volume of an object represented as point set surface into cubes, giving a non-overlapping discrete representation of the domain, on which efficient numerical integration schemes can be employed. The octants serve as the basic unit from which to construct the patches and allow the efficient determination of patch interactions. In the following subsection, we will describe the use of the octree structure as the basic building block to help us define our mesh-free patches and integration cells.

4.1.1 Octree-Based Volumetric Node Placement

An octree structure can be defined by enclosing the object domain of interest Ω in a cube which represents the root of the octree, and then subdividing the cube into eight octants of the root by bisection along all three directions. The octants are recursively subdivided to whichever levels are desired. Note that the terminal level used for our node placement does not need to coincide with the terminal level of the MPU implicit surface construction. Actually, in our implementation, the size of the terminal octant used for our volumetric node placement (for mesh-free simulation) is much larger than the terminal octant used for MPU implicit surface reconstruction because the surface point density is much larger compared to the volumetric node density. Figure 2 (b)(c) shows the octree-based discretization for the MPU implicit surface construction and volumetric node placement. We restrict the octree to be a one level adjusted octree, where the level difference of all terminal octants and their face and edge neighbors is no more than one. This restriction can facilitate the automatic satisfaction of patch covering condition (Equation (8)) as we will discuss later.

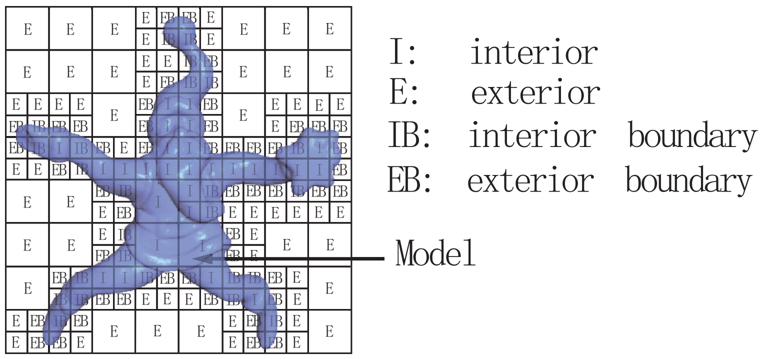

Fig. 3 The definition of interior, exterior, interior boundary, and exterior boundary octants for mesh-free simulation.

Since we already have the implicit surface representation of the object, we can easily classify each terminal octants as interior (I) octants O_I, exterior (E) octants O_E, and boundary (B) octants O_B (see Figure 3). Interior octants are those that are fully embedded in the interior of the geometric domain Ω. Exterior octants are those that are located totally outside of Ω, and boundary octants are those that are intersected by the boundary of Ω. The boundary octants are further classified into interior boundary (IB) O_{IB} and exterior boundary (EB) O_{EB} octants. The simple rule is that the centroid of an IB octant is located within the domain, whereas the centroid of an EB octant is located outside the domain. After the geometric classification, we can place a volumetric node (for mesh-free dynamics) at the center of each interior (I) and boundary (IB, EB) octant. For an EB octant, the node should be displaced by projecting from its center onto the implicit surface to ensure that each node resides in

Ω. Let octant $O_i \in O_I \cup O_B$ and node i reside in O_i, the open cover associated with node i is a cube of size $\alpha \cdot size(O_i)$ centered around node i (see Figure 4 (a)). Both the volumetric nodes and their open cover regions are necessary constituents for mesh-free dynamics.

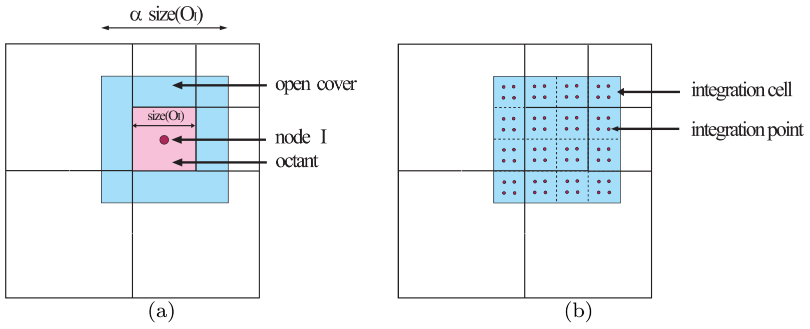

Fig. 4 (a) The definition of open cover $\{\Omega_I\}$ regions based on the octree structure for mesh-free patches; (b) The interaction between open covers and integration cells, and the integration points for Gaussian quadrature.

The open cover construction based on terminal octants can provide the structure needed to perform efficient neighboring search and patch intersection test. It has been proved in [15] that by choosing a suitable size for α, the validity of the open cover can be guaranteed a priori. For example, for a linear basis $\mathbf{p}(\mathbf{x})^T_{(m=4)} = \{1, x, y, z\}$, any point in the domain will be covered by at least 4 patches if we choose α to be 3. The generation of an octree is much more efficient than a finite element mesh in practice. Furthermore, the octree allows refinement of the discretization in areas of singularities if necessary (e.g., near the crack surface).

4.1.2 Octree-Based Gaussian Integration for Matrix Assembly

In order to assemble the entries of the system matrices, such as the mass matrix or stiffness matrix, we need to integrate over the problem domain. This can be performed through numerical techniques such as Gaussian quadrature, using the underlying integration cells. The integration cells can be totally independent of the arrangement of nodes. The integration cells are used merely for the integration of the system matrices but not for field value interpolation. In our octree-based discretization scheme, since the terminal octants do not overlap (except on their shared boundaries), we can further subdivide the terminal octants O_I and O_B into smaller cells and use them as the integration cells (see Figure 4 (b)). There may exist some integration cells that do not entirely belong to the analysis domain. We can easily separate the portion of the cell which lies outside of the domain by evaluating the implicit function (used for representing the surface distance field). The creation of the open cover and the integration cells, as described here, eliminates any global searching for members of the open cover during matrix assembly and time integration. With the prior knowledge of the value α and utilizing the direct face neighbor links, all patches covering a integration point $\mathbf{x} \in \Omega$ can be found in $O(1)$ time.

4.2 Parameterization-Based Quadtree Discretization for Thin-Shell Models

The implicit surface representation mentioned above is essentially volumetric embedding. It does not admit a natural, 2-dimensional domain for the effective analysis of point-sampled surfaces. As a result, simulating physical behavior of point-sampled thin-shells becomes rather difficult, compared with the direct simulation of volumetric models. Although the volumetric simulation mechanism is both natural and intuitive, it is essentially plagued by the additional computational burden in both time and space. In many real-world applications, such as the deformation of the wings of the gargoyle, or even open surfaces like a plate, thin-shell simulation is much better than volumetric one, since one dimension of the surface body is much smaller than the other two, and the volumetric simulation may fail if the neighboring volumetric nodes are arranged in degenerate locations, such as a plane. To combat the deficiency associated with the dimensional increase, we can utilize the global conformal parameterization [12] of the point-sampled surfaces, to obtain a natural 2-dimensional parametric domain for simulation node placement and numerical integration. Figure 5 shows the global conformal parameterization for the David head surface model. Figure 6 shows the simulation nodes (and their support radii) placed on the Moai surface model and it corresponding parametric domain.

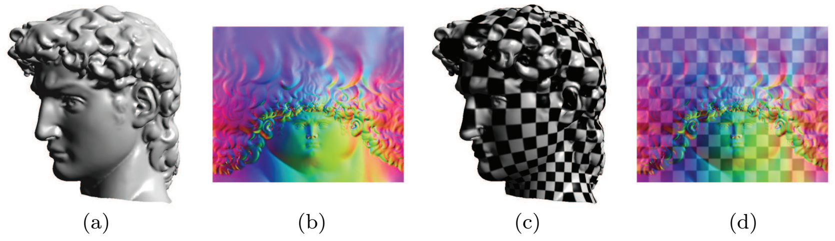

Fig. 5 The global conformal parameterization (b) for the David head model (a). The checkerboard texture on the parametric domain (d) can be mapped onto the surface (c).

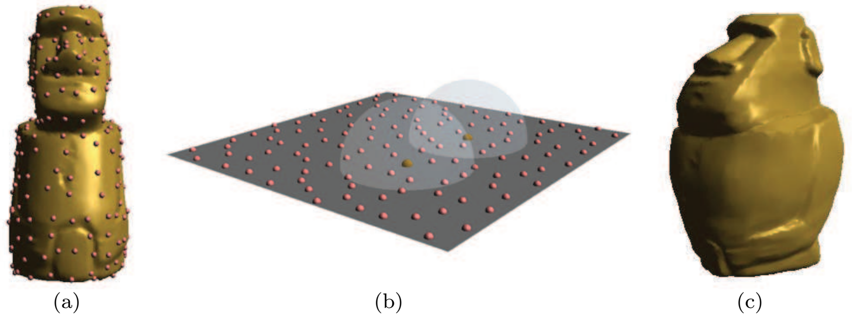

Fig. 6 The simulation nodes placed on the Moai surface model (a) and its parametric domain (b). The larger translucent white hemispheres represent the support radii of two of the simulation nodes. (c) shows the deformed Moai surface using the set of simulation nodes.

One important advantage of meshless method is the flexibility of the sampling pattern. In the thin-shell simulation, it would be more desirable to have a initial sampling scheme such that the sampling nodes are uniformly distributed on the manifold surface (Figure 8 (a)). Similar to the idea of using octree structure to facilitate the volumetric sampling of the volumetric space in Section 4.1, we utilize a quadtree structure on the parametric domain. The subdivision depth of the quadtree is dependent on the conformal factor λ. Suppose the size of the quadtree cell is l. If λl is larger than a threshold (i.e. the surface patch corresponding to the cell is still large enough), we keep subdividing the cell into 4 child-cells. We place the sampling nodes based on the quadtree discretization of the parametric plane. Since the conformal factors in u and v directions are equivalent, we can choose simple "symmetric" supporting region (such as squares, or disks) for each simulation nodes. In our implementation, we use square-shaped supporting region for the ease of performing numerical integrations. For a node i reside in quadtree cell Q_i, its supporting region is a square of size $\eta \cdot size(Q_i)$ centered around node i. We restrict the quadtree to be a one-level adjusted quadtree, where the level difference of all terminal cells and their edge neighbors is no more than one. This restriction can facilitate the automatic satisfaction of patch covering condition in order to make the moment matrix invertible. It has been proved in [15] that by choosing a suitable size for η, the validity of the supporting region can be guaranteed a priori. For example, for a linear basis $\mathbf{p}(\mathbf{x})^T = [1, \theta^1, \theta^2]$, any point in the domain will be covered by at least 3 patches if we choose η to be 3. The supporting region construction based on terminal quadtree cells can provide the structure needed to perform efficient neighboring search and patch intersection test. The quadtree cells can be also utilized as integration cells to perform numerical integration similar to the volumetric case in Section 4.1.2.

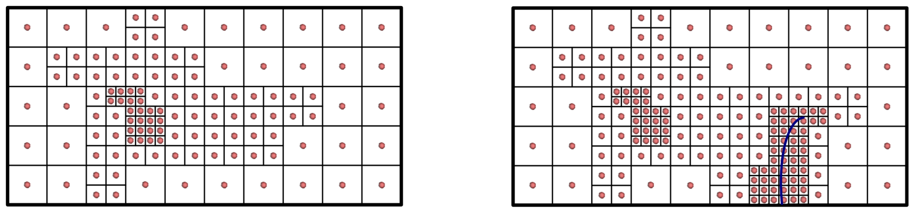
(a) (b)

Fig. 7 (a) The quadtree structure on the parametric plane for placing simulation nodes; (b) the dynamic re-sampling near the crack line (blue curve).

In the parameterization stage, the seams between different parametric patches are represented by additional point set curves, which are generated by tracing integral curves. These additional point set curves are only used in the parameterization step, i.e., they are not added to the original point set surface after the parameterization. In the simulation stage, the parametric seams are maintained in a table indicating the connection correspondence of different parametric patches. The simulation nodes are not duplicated on the patch boundaries. The support of the shape function associated with each simulation node is not restricted to the parametric patch where it resides. In fact, the support can be expanded to other parametric patch if the node is close to the boundary of its patch (see Figure 8 (c)). So the behaviors of the nodes across the boundaries are consistent with non-boundary nodes without any unnatural artifact.

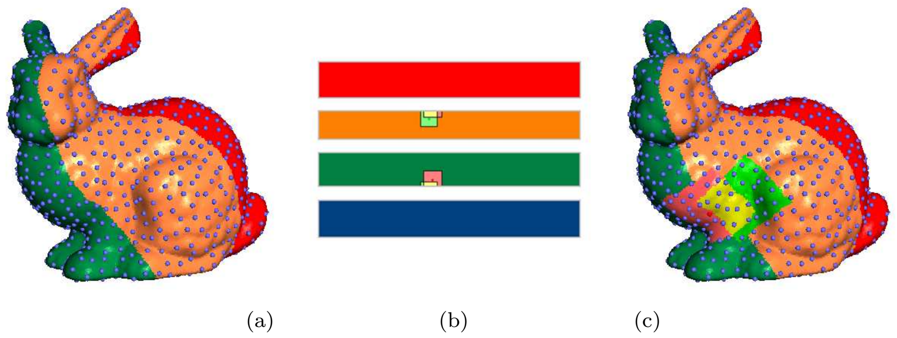

Fig. 8 (a) Uniform sampling nodes on the surface; (b) the 4 parametric planes; (c) the supporting regions of two simulation nodes are shown in pink and green; their overlapping region is shown in yellow.

When a crack is generated in a body, the dependent variables (e.g. the displacements, etc.), must be discontinuous across the crack. Furthermore, the support of the nodes affected by the discontinuities need to be modified accordingly to incorporate the proper behavior of the shape functions and its derivatives. Similar to the approach of [27], we use the transparency criterion proposed in [25] to allow partial interaction of nodes in the vicinity of the crack front. Note that all these operations can be easily performed on the 2D parametric domain. We need to perform dynamic up-sampling (node insertion) during the crack simulation, in order to maintain numerical stability even for an initially adequately sampled model. New simulation nodes need to be inserted in the vicinity of crack lines. We take the same criterion as [27] to determine under-sampling at each node based on transparency weights. If the transparency weights becomes too small due to a nearby crack line, we subdivide the quadtree cell associated with the simulation node into 4 child-nodes (see Figure 7 (b)).

5 Dynamic Deformation System

In our meshless approximation, the motion parameters of the material point $\mathbf{x}$, i.e., the displacements $\mathbf{u}$, velocity $\dot{\mathbf{u}}$, and acceleration $\ddot{\mathbf{u}}$, can be approximated by using the moving least squares shape functions $\phi_I(\mathbf{x})$ as:

$$\mathbf{u}(\mathbf{x},t) = \sum_I \phi_I(\mathbf{x})\mathbf{u}_I(t), \tag{14}$$

$$\dot{\mathbf{u}}(\mathbf{x},t) = \sum_I \phi_I(\mathbf{x})\dot{\mathbf{u}}_I(t),$$

$$\ddot{\mathbf{u}}(\mathbf{x},t) = \sum_I \phi_I(\mathbf{x})\ddot{\mathbf{u}}_I(t).$$

Note that $\mathbf{u}_I$, $\dot{\mathbf{u}}_I$, and $\ddot{\mathbf{u}}_I$ are not the nodal values of displacements, (velocities, etc.), but rather nodal parameters without a direct physical interpretation, because the shape functions $\phi_I(\mathbf{X})$ produce approximation, not interpolation of the field values. The partial derivatives with respect to the referencing coordinates x_k can be obtained simply as:

$$\mathbf{u}_{,k}(\mathbf{x}, t) = \sum_I \phi_{I,k}(\mathbf{x})\mathbf{u}_I(t).$$

5.1 Dynamic Elastic Solids

We use the *Euler-Lagrange* equations for the elastic deformation of solid models:

$$\frac{d}{dt}(\frac{\partial T(\dot{\mathbf{u}})}{\partial \dot{\mathbf{u}}}) + \mu\dot{\mathbf{u}} + \frac{\partial V(\mathbf{u})}{\partial \mathbf{u}} = \mathbf{F}_{ext}, \tag{15}$$

where the kinetic energy T and elastic potential energy V are functions of $\dot{\mathbf{u}}$ and $\mathbf{u}$, respectively. The term $\mu\dot{\mathbf{u}}$ is the generalized dissipative force, and $\mathbf{F}_{ext}$ is a generalized force arising from external body forces, such as gravity.

The kinetic energy of the moving body can be expressed as:

$$T = \frac{1}{2}\int_\Omega \rho(\mathbf{x})\dot{\mathbf{u}} \cdot \dot{\mathbf{u}} d\Omega = \frac{1}{2}\sum_{I,J} M^{IJ}\dot{\mathbf{u}}_I \cdot \dot{\mathbf{u}}_J, \tag{16}$$

where $\rho(\mathbf{x})$ is the mass density of the body, and $M^{IJ} = \int_\Omega \rho(\mathbf{x})\phi_I(\mathbf{x})\phi_J(\mathbf{x})d\Omega$. Then we can have:

$$\frac{d}{dt}(\frac{\partial T(\dot{\mathbf{u}})}{\partial \dot{\mathbf{u}}}) = \mathbf{M}\ddot{\mathbf{u}}, \tag{17}$$

where the matrix $\mathbf{M}$ composed of the elements $\mathbf{M}^{IJ}$ is called the *mass matrix*.

The elastic potential energy of a body can be expressed in terms of the *strain tensor* and *stress tensor*. The strain is the degree of metric distortion of the body. A standard measure of strain is Green's strain tensor:

$$\epsilon_{ij} = \frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \delta_{kl}\frac{\partial u_k}{\partial X_i}\frac{\partial u_l}{\partial X_j}. \tag{18}$$

Forces acting on the interior of a continuum appear in the form of the *stress tensor*, which is defined in terms of strain:

$$\tau_{ij} = 2G\{\frac{\nu}{1-2\nu}tr(\epsilon)\delta_{ij} + \epsilon_{ij}\}, \tag{19}$$

where $tr(\epsilon) = \sum_{ij}\delta_{ij}\epsilon_{ij}$. The constant G is called the *shear modulus*, which determines how strongly the body resists deformation. The coefficient ν, called *Poisson's ratio*, determines the extent to which strains in one direction are related to those perpendicular to it. This gives a measure of the degree to which the body preserves volume. The elastic potential energy $V(\mathbf{u})$ is given by the formula:

$$V = G\int_\Omega \{\frac{\nu}{1-2\nu}tr^2(\epsilon) + \sum_{ijkl}\delta_{ij}\delta_{kl}\epsilon_{ik}\epsilon_{jl}\}d\Omega, \tag{20}$$

By combining the above Equations (14), (18), and (23), we can formulate the derivatives of V (with respect to $\mathbf{u}$) as polynomial functions of $\mathbf{u}$, the coefficients of which are integrals that can be pre-computed using the techniques mentioned in Section 4.1.2.

5.2 Dynamic Elastic Thin-Shells

For any point-sampled surface, if we assume that one dimension (i.e. the thickness), of the surface body is significantly smaller than the other two dimensions, we can consider the point-sampled surface as a thin-shell. In the Kirchhoff-Love thin shell framework, the deformation of the surface body is fully described by the deformation of the middle surface represented by point-samples. Let φ denote the position of a point on the middle surface of the shell, and let $\mathbf{a}_3$ be the unit director vector which is normal to the shell surface. Given the global parameterization of the point-sampled surfaces, we can describe the positions of any material point in the reference (denoted $\bar{\mathbf{r}}$) and deformed (denoted $\mathbf{r}$) configurations of the shell by:

$$\bar{\mathbf{r}}(\theta_1, \theta_2, \theta_3) = \bar{\varphi}(\theta_1, \theta_2) + \theta_3 \bar{\mathbf{a}}_3(\theta_1, \theta_2), \tag{21}$$

$$\mathbf{r}(\theta_1, \theta_2, \theta_3) = \varphi(\theta_1, \theta_2) + \theta_3 \mathbf{a}_3(\theta_1, \theta_2), \tag{22}$$

where θ_1 and θ_2 are parameters of the point-sampled middle surface, and θ_3 $(-\frac{h}{2} \leq \theta_3 \leq \frac{h}{2})$ is in the thickness direction.

The precise form of the membrane and bending strain and stress matrices are given in [6]. Similar to the elastic solids, we use the *Euler-Lagrange* equation (24) for our dynamic thin-shell deformation. The kinetic energy of the moving thin-shell is defined similar to the elastic solids in equation (16), except that the integration domain Ω is the global conformal parametric domain. The elastic potential energy V is given by the formula:

$$V = \int_{\Omega} [\frac{Eh}{1-\nu^2} \alpha^T \tilde{\mathbf{H}} \alpha + \frac{Eh^3}{12(1-\nu^2)} \beta^T \tilde{\mathbf{H}} \beta] d\Omega. \tag{23}$$

where α and β are the membrane and bending strains respectively, $\tilde{\mathbf{H}}$ is the standard constitutive matrix, the constant E is *Young's modulus*, and the coefficient ν is *Poisson's ratio*. More specific derivations can be found in [6].

5.3 Solving the Dynamic System

The system of ordinary differential equations which results from the application of the Element-free Galerkin discretization of the spatial domain can either be integrated directly, or analyzed by *mode superposition*. That is, the time dependent solution can be expressed as the superposition of the natural (or resonant) modes of the system. In the following section, we will briefly introduce some basics of *Modal Analysis*. More detailed discussions can be found elsewhere [28,14,13,5].

5.3.1 Basics of Modal Analysis:

Consider the discretized *Euler-Lagrange* equations for elastic deformation:

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{F}(t) \tag{24}$$

where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ are the mass, damping and stiffness matrices, respectively, $\mathbf{F}$ is the external load vector and $\mathbf{u}(t)$ is the vector of nodal displacements. Under the commonly adopted *Rayleigh damping* assumption, we can replace the damping matrix with $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$, where α and β are weighting coefficients. For linear elasticity models, both

$\mathbf{M}$ and $\mathbf{K}$ are constants. Let the columns of $\mathbf{\Psi}$ be the solution to the generalized eigenvalue problem $\mathbf{Kx} = \lambda\mathbf{Mx}$, and $\mathbf{\Lambda}$ be the diagonal matrix of eigenvalues, then equation (24) can be transformed to:

$$\ddot{\mathbf{z}} + (\alpha\mathbf{I} + \beta\mathbf{\Lambda})\dot{\mathbf{z}} + \mathbf{\Lambda z} = \mathbf{\Psi}^T\mathbf{F}, \tag{25}$$

where $\mathbf{z} = \mathbf{\Psi}^{-1}\mathbf{u}$ is the vector of modal amplitudes, and $\mathbf{\Psi}$ is called *modal displacement matrix* whose i-th column represents the i-th mode shape (see Figure 9). The decoupled ODEs in Equation (25) can be computed independently and combined by linear superposition. The computational loads can be further reduced by removing modes that are too stiff to be observed (corresponding to higher eigenvalues). So we can take only l dominant columns of $\mathbf{\Psi}$, to reduce the amount of computation significantly.

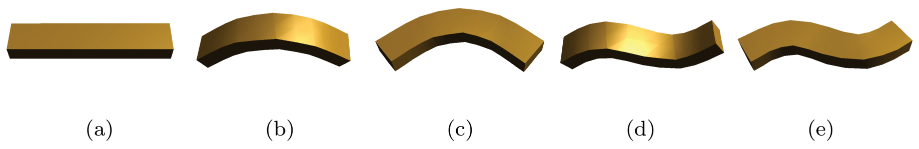

Fig. 9 (a) is the beam model before deformation; (b)-(e) show the 6^{th}, 7^{th}, 8^{th}, and 11^{th} mode shapes, respectively.

5.3.2 Modal Warping for Rotational Deformation:

Our Meshless Modal Analysis framework is build upon the *Modal Warping* technique proposed by Choi and Ko [5]. Their innovative approach tracks the local rotations that occur during the deformation based on the infinitesimal rotation tensor, and warps the pre-computed modal basis in accordance with the local rotations of the mesh nodes. For the space limit, we only briefly introduce their general ideas here. More specific technical details and proofs can be found in [5].

Considering an infinitesimal deformation with displacement $\mathbf{u}$, the rotation tensor is defined as:

$$\omega = \frac{1}{2}(\nabla \times \mathbf{u})\times = \mathbf{w}\times, \tag{26}$$

where $\nabla \times \mathbf{u}$ is the curl of the displacement, $\mathbf{w}\times$ denotes the standard skew-symmetric matrix of vector $\mathbf{w}$. Here $\mathbf{w} = \frac{1}{2}(\nabla \times \mathbf{u})$ can be considered as a rotation vector that causes the rotation by angle $\|\mathbf{w}\|$ around the unit axis $\mathbf{w}/\|\mathbf{w}\|$. In the Modal Analysis setting, the rotation vector can be expressed in terms of the modal amplitude $\mathbf{z}$:

$$\mathbf{w}(\mathbf{x}) = \frac{1}{2}(\nabla\times)\mathbf{\Phi}(\mathbf{x})\mathbf{\Psi z} \tag{27}$$

where $\mathbf{\Phi}(\mathbf{x})$ is the vector of MLS shape functions evaluated at position $\mathbf{x}$.

The basic idea of the *Modal Warping* approach is to embed a local coordinate frame at each simulation node (see Figure 10). The rotation matrix $\mathbf{R}_i$ of the local coordinate frame associated with node i can be computed from its rotation vector $\mathbf{w}_i$. For a general non-linear elastic deformable model, the stiffness matrix $\mathbf{K}(\mathbf{u})$ is not a

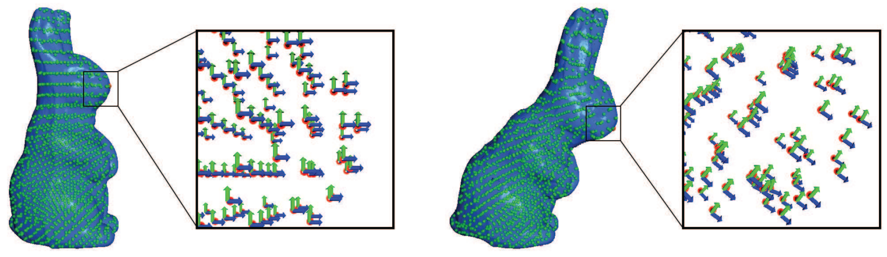
(a) (b)

Fig. 10 A solid rabbit model before (a) and after (b) deformation. The green spheres are the discretized simulation nodes inside the rabbit surface. The local coordinate frames associated with the simulation nodes are rotated during the deformation.

constant. In order to apply the linear Modal Analysis method, it has been shown in [5] that the non-linear *Euler-Lagrangian* equations

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u})\mathbf{u} = \mathbf{F} \tag{28}$$

can be approximated using the displacement $\mathbf{u}^L$ measured from each local orientation frame:

$$\mathbf{M}\ddot{\mathbf{u}}^L + \mathbf{C}\dot{\mathbf{u}}^L + \mathbf{K}\mathbf{u}^L = \mathbf{R}^T\mathbf{F} \tag{29}$$

where $\mathbf{R} = [\delta_{ij}\mathbf{R}_i]$ is the block diagonal rotation matrix for all the nodes. Actually there are two basic assumptions to guarantee the validity of this approximation (please refer to [5] for the details and proofs). And we found that these assumptions can be directly applied to our meshless setting without influencing its validity. Using modal decomposition: $\mathbf{u}^L(t) = \mathbf{\Psi}\mathbf{z}^L(t)$, the linear elastodynamic equation (29) for $\mathbf{u}^L$ can be reduced to a set of decoupled ODEs:

$$\ddot{\mathbf{z}}^L + \mathbf{C}_z\dot{\mathbf{z}}^L + \mathbf{K}_z\mathbf{z}^L = \mathbf{\Psi}^T(\mathbf{R}^T\mathbf{F}). \tag{30}$$

where $\mathbf{C}_z = (\alpha\mathbf{I} + \beta\mathbf{\Lambda})$ and $\mathbf{K}_z = \mathbf{\Lambda}$ are both diagonal matrices. We solve the above decoupled ODEs using implicit time integration. We take an approach similar to [1] by making a first-order approximation of the total force at the next time step, to get the following linear system:

$$\begin{cases} \Delta\mathbf{z} = h(\dot{\mathbf{z}}_0 + \Delta\dot{\mathbf{z}}) \\ \Delta\dot{\mathbf{z}} = h(\mathbf{F}_0 - \mathbf{K}_z(\mathbf{z}_0 + \Delta\mathbf{z}) - \mathbf{C}_z(\dot{\mathbf{z}}_0 + \Delta\dot{\mathbf{z}})) \end{cases}$$

where h is the size of the time step, $\mathbf{z}_0$ and $\dot{\mathbf{z}}_0$ are the current modal amplitude and velocity, and $\Delta\mathbf{z}$ and $\Delta\dot{\mathbf{z}}$ are their expected change in the next time step. By regrouping, we obtain

$$\mathbf{A}_z\Delta\dot{\mathbf{z}} = \mathbf{b}_z \tag{31}$$

where

$$\mathbf{A}_z = \mathbf{I} + h\mathbf{C}_z + h^2\mathbf{K}_z$$

and

$$\mathbf{b}_z = h\left(\mathbf{F}_0 - \mathbf{K}_z\mathbf{z}_0 - (\mathbf{C}_z + h\mathbf{K}_z)\dot{\mathbf{z}}_0\right)$$

Note that $\mathbf{A}_z$ is a diagonal matrix, which makes equation (31) to be solved efficiently.

5.3.3 Manipulation constraints

In order for the users to interact with the simulated objects, position and orientation constraints are important and must be enforced. In general, the MLS shape functions lack the Kronecker delta function property and result in $\mathbf{u}(\mathbf{x}_I) \neq \mathbf{u}_I$. The position and orientation constraints ($\mathbf{C}_p$ and $\mathbf{C}_o$, respectively) can be formulated as:

$$\mathbf{C}_p(\mathbf{x}_c) = \mathbf{\Phi}(\mathbf{x}_c)\mathbf{u}(t) - \mathbf{d}_c(t) = 0$$

$$\mathbf{C}_o(\mathbf{x}_c) = \frac{1}{2}(\nabla\times)\mathbf{\Phi}(\mathbf{x}_c)\mathbf{u}(t) - \mathbf{w}_c(t) = 0$$

where $\mathbf{x}_c$ is the constrained position of the object, $\mathbf{d}_c(t)$ is the desired displacement, and $\mathbf{w}_c(t)$ is the desired orientation, which are known a priori. If we express both the position and orientation constraints in terms of the modal amplitude $\mathbf{z}$, they can be simply written as:

$$\mathbf{C} = \mathbf{A}_c\mathbf{z} - \mathbf{b}_c = 0 \tag{32}$$

where $\mathbf{A}_c$ is a $k \times n$ constraint matrix (k is the number of constraints), and each row of $\mathbf{A}_c$ represents a linear constraint on $\mathbf{z}$, and the vector $\mathbf{b}_c$ represents the values of these constraints. The constraint condition (32) can be integrated into the system equation (31) by Lagrange multipliers. In our implementation, we replace the constraint equation $\mathbf{C} = 0$ by the damped second-order equation $\ddot{\mathbf{C}} + 2\eta\dot{\mathbf{C}} + \gamma^2\mathbf{C} = 0$, where η and γ are stabilization factors [20]. So we can obtain the constrained equations of motion:

$$\begin{bmatrix} \mathbf{A}_z & \mathbf{A}_c^T \\ \mathbf{A}_c & 0 \end{bmatrix} \begin{bmatrix} \Delta\dot{\mathbf{z}} \\ \lambda h \end{bmatrix} = \begin{bmatrix} \mathbf{b}_z \\ h(-2\eta\dot{\mathbf{C}} - \gamma^2\mathbf{C}) \end{bmatrix}. \tag{33}$$

Since both the number of selected modes l and the number of constraints k are typically small ($l \leq 128$, $k \leq 20$ in all of our examples), equation (33) could be solved in real-time.

6 Experimental Results

The meshless simulation and rendering parts of our system are implemented on a Windows XP PC with dual Intel Xeon 2.8GHz CPUs, 2.0GB RAM, and an nVidia GeForce Fx 5900 Ultra GPU. The entire point-based rendering pipeline is built upon Pointshop3D [32]. We have conducted extensive experiments on various point-sampled surface data sets, for both solid and thin-shell simulations.

Table 1 shows the model statistics and performance data for our meshless solid simulation based on the modal warping technique. For all the data sets in the elastic solid simulation, the MLS pre-computation for the system matrices takes less than 10 minutes, while the modal decomposition takes less than 1 minute. Figure 11 shows the deformation of the Santa Claus model based on users' manipulation constraints on its hands and feet. Figure 12 shows the deformation of the balljoint model under users' manipulation with the bottom fixed.

Table 2 shows the performance data for our meshless thin-shell simulation based on global conformal parameterization. We use the implicit time integration method for the thin-shell simulation, which has been demonstrated to be very stable. The initial numbers of nodes in the examples of gargoyle, bunny, rocker arm, and Iphigenie simulations are only around 500. The elastic deformations of the gargoyle (Figure 13),

Table 1 The model statistics and performance data for the meshless elastic solid simulation based on modal warping.

Model	Points	Nodes	Modes	Simulation
Beam	5,634	1,008	64	0.013 s/f
Balljoint	137,062	357	64	0.026 s/f
Rabbit	67,038	1,251	64	0.019 s/f
Santa	75,781	1,150	128	0.028 s/f

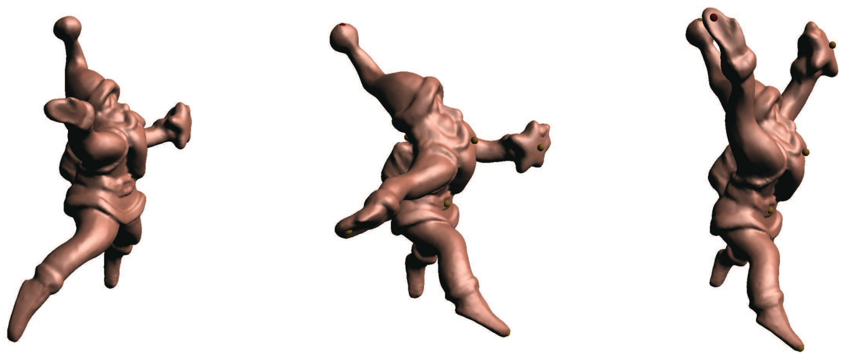

Fig. 11 A solid Santa model before (a) and after (b,c) elastic deformation.

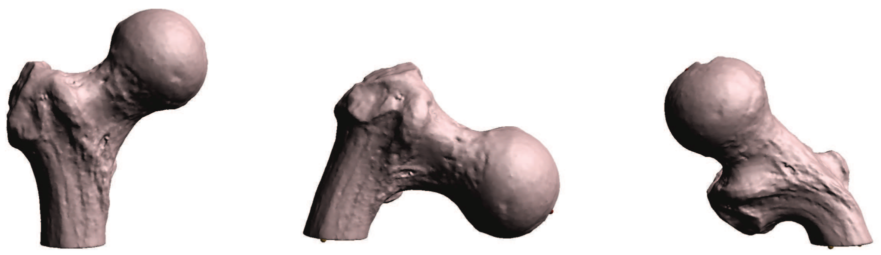

Fig. 12 A solid balljoint model before (a) and after (b,c) elastic deformation.

bunny (Figure 14), and rocker arm (Figure 15) are simulated in real-time. Figure 16 shows the global conformal parameterization and explosion simulation of the Iphinenie model. The computational load at each time step of the fracture propagation is much higher than pure elastic deformation. We need to perform transparency tests for the integration points, surface points, and simulation nodes in the neighborhood of new crack lines. And new simulation nodes need to be inserted if necessary. We have to update the mass and stiffness matrices at each time step to accomodate these changes.

7 Conclusion and Future Work

Large-scale point-sampled geometry is becoming ubiquitous in graphics, visualization, and geometric information processing, due to the rapid advancement of the 3D scanning

Table 2 The model statistics and performance data for the meshless elastic thin-shell simulation based on global conformal parameterization.

Model	Points	Parameterization	Nodes (Initial/Final)	Simulation
Rockerarm	40,000	88.63 s	405/405	0.020 s/f
Gargoyle	40,002	62.09 s	520/520	0.026 s/f
Bunny	50,002	99.85 s	440/440	0.022 s/f
Iphigenie	200,219	866.45 s	420/650	1.2 s/f

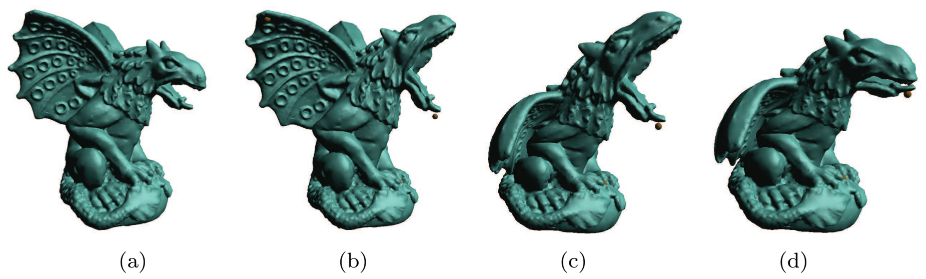

Fig. 13 A thin-shell gargoyle model before (a) and after (b-d) elastic deformation.

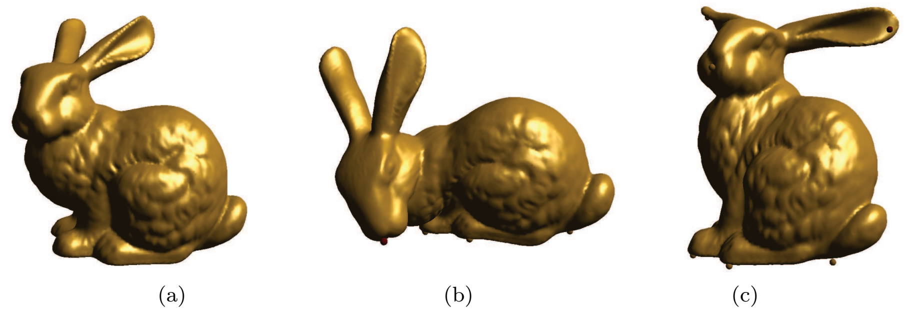

Fig. 14 A thin-shell bunny model before (a) and after (b,c) elastic deformation.

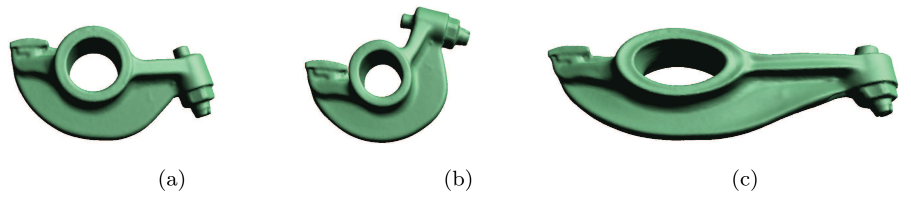

Fig. 15 A thin-shell rocker arm model before (a) and after (b,c) elastic deformation.

devices. Developing new computational techniques for point-centered digital processing has become a common and long-term mission in computer graphics. In this paper, we present our recent research results in meshless simulation of deformable shells and solids. Our meshless framework is build upon continuum mechanics, which provide a rigorous foundation to simulate the physical behavior of the modeled point geometry. We presented several key computational techniques: (1) Moving Least Squares for func-

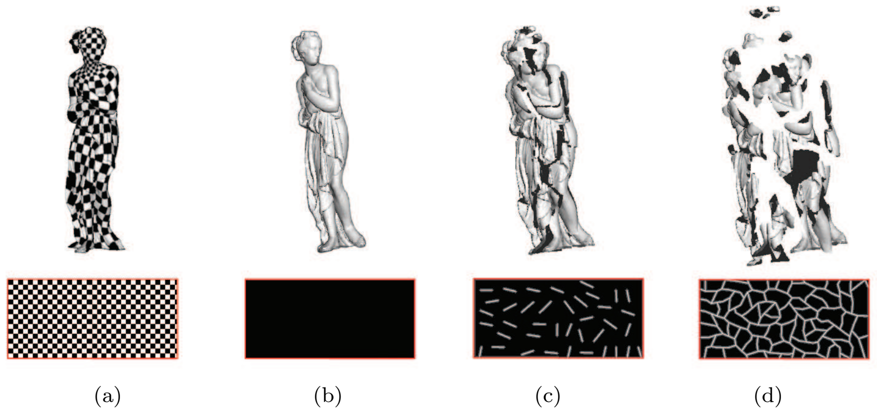

Fig. 16 (a) The global conformal parameterization of the Iphigenie model. (b-d) The meshless thin-shell fracture simulation based on the parameterization. The bottom parts show the crack pattern in the parametric domain.

tional approximation and interpolation using discrete nodes; (2) hierarchical meshless discretization for shells and solids; (3) Modal Warping for real-time simulation of large deformations.

There are many avenues for possible future work, including exploring the theoretical foundation of the meshless framework, designing efficient and accurate computational algorithms, developing new functionalities, e.g. integrating haptic interaction between human beings and virtual environments. Some of the most exciting applications of HCI and haptic feedback have been found in surgical simulation, including a variety of medical procedures, e.g. limb surgery, plastic surgery, gastrointestinal endoscopy, etc. Traditionally mesh-based FEM is used to simulate real-time visual and haptic rendering of soft tissues. A potential solution to the remeshing problems faced by the FEM techniques is to use meshless methods articulated in this paper. Simulating surgical operations such as cutting and coagulation requires frequently updating the organ mesh structure which is a very challenging process to be implemented robustly without numerical singularities. Surgical cutting simulation is a simplified version of the fracture simulation, since the cutting path is human-guided. However, surgical simulations pose more critical time constraints since they need to provide timely sensory information for the user in order to achieve immersive realism. A balance is needed to ensure the simulation is fast enough for generating real-time feedbacks whilst the level of realism is reasonably acceptable. Numerical methods, including the solvers used in our meshless methods, can be very computationally expensive. Fortunately, many such numerical techniques can be parallelized in order to accelerate the computation of relevant quantities. In particular, the hierarchical structure (quadtree and octree) used in our thin-shell and volumetric simulations can facilitate the parallelization by computing strains and other relevant quantities locally and by assembling the results globally. Graphics processing unit (GPU) has seen fast developing in recent years. More and more researchers have tried to use the GPU to solve general-purpose problems. Some GPU-based mesh deformation algorithms have already been proposed [31]. However, the GPU pipeline is specifically designed for mesh-based computations. Using the parallel computing ability of the GPU to accelerate meshless simulation is an interesting direction for the future work.

References

1. Baraff D, and Witkin A. Large steps in cloth simulation. Proceedings of SIGGRAPH'98, pp. 43-54, 1998.
2. Belytschko T, Lu Y Y, and Gu L. Element free galerkin methods. International Journal for Numerical Methods in Engineering, 37: 229-256, 1994.
3. Belytschko T, Lu Y Y, and Gu L. Fracture and crack growth by element-free galerkin methods. Modeling Simulations for Materials Science and Engineering, 2: 519-534, 1994.
4. Belytschko T, Krongauz Y, Organ D, Fleming M, and Krysl P. Meshless methods: an overview and recent developments. Computer Methods in Applied Mechanics and Engineering, 139: 3-47, 1996.
5. Choi M G, and Ko H S. Modal warping: real-time simulation of large rotational deformation and manipulation. IEEE Transactions on Visual Computing and Graphics, 11(1): 91-101, 2005.
6. Cirak F, Ortiz M, and Schröder P. Subdivision surfaces: a new paradigm for thin-shell finite element analysis. International Journal for Numerical Methods in Engineering, 47(12): 2039-2072, 2000.
7. DeRose T, Kass M, and Truong T. Subdivision surfaces in character animation. Proceedings of SIGGRAPH'98, pp. 85-94, 1998.
8. Desbrun M, and Cani M P. Animating soft substances with implicit surfaces. Proceedings of SIGGRAPH'95, pp. 287-290, 1995.
9. Farin G. Curves and Surfaces for CAGD. Morgan-Kaufmann, 5th edition, 2001.
10. Guo X, Hua J, and Qin H. Scalar-function-driven editing on point set surfaces. IEEE Computer Graphics and Applications, 24(4): 43-52, 2004.
11. Guo X, and Qin H. Real-time meshless deformation. Computer Animation and Virtual Worlds, 16(3-4):189-200, 2005.
12. Guo X, Li X, Bao Y, Gu X, and Qin H. Meshless thin-shell simulation based on global conformal parameterization. IEEE Transactions on Visualization and Computer Graphics, 12(3): 375-385, 2006.
13. Hauser K, Shen C, and O'Brien J. Interactive deformation using modal analysis with constraints. Proceedings of Graphics Interface, pp. 247-255, 2003.
14. James D, and Pai D. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. Proceedings of SIGGRAPH'02, pp. 582-585, 2002.
15. Klaas O, and Shephard M. Automatic generation of octree-based three-dimensional discretizations for partition of unity methods. Computational Mechanics, 25: 296-304, 2000.
16. Lancaster P, and Salkauskas K. Curve and surface fitting: an introduction. Academic Press, London, 1986.
17. Levoy M, Pulli K, Curless B, Rusinkiewicz S, Koller D, Pereira L, Ginzton M, Anderson S, Davis J, Ginsberg J, Shade J, and Fulk D. The digital Michelangelo project: 3d scanning of large statues. Proceedings of SIGGRAPH'00, pp. 131-144, 2000.
18. Li S, and Liu W K. Meshfree particle methods and their applications. Applied Mechanics Review, 54: 1-34, 2002.
19. Malladi R, Sethian J A, and Vemuri B C. Shape modeling with front propagation: A level set approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17: 158-175, 1995.
20. Metaxas D, and Terzopoulos D. Dynamic deformation of solid primitives with constraints. Proceedings of SIGGRAPH'92, pp. 309-312, 1992.
21. Müller M, Keiser R, Nealen A, Pauly M, Gross M, and Alexa M. Point-based animation of elastic, plastic, and melting objects. Proceedings of ACM SIGGRAPH/ Eurographics Symposium on Computer Animation, 2004.
22. Müller M, Heidelberger B, Teschner M, and Gross M. Meshless deformations based on shape matching. ACM Transactions on Graphics, 24(3):471-478, 2005
23. O'Brien J F, and Hodgins J K. Graphical modeling and animation of brittle fracture. Proceedings of SIGGRAPH'99, pp. 137-146, 1999.
24. Ohtake Y, Belyaev A, Alexa M, Turk G, and Seidel H-P. Multi-level partition of unity implicits. Proceedings of SIGGRAPH'03, pp. 463-470, 2003.

25. Organ D, Fleming M, Terry T, and Belytschko T. Continuous meshless approximations for nonconvex bodies by diffraction and transparency. Computational Mechanics, 18: 1-11, 1996.
26. Pauly M, Keiser R, Kobbelt L, and Gross M. Shape modeling with point-sampled geometry. Proceedings of SIGGRAPH'03, pp. 641-650, 2003.
27. Pauly M, Keiser R, Adams B, Dutré P, Gross M, and Guibas L. Meshless animation of fracturing solids. Proceedings of SIGGRAPH'05, pp. 957-964, 2005.
28. Pentland A, and Williams J. Good vibrations: model dynamics for graphics and animation. Proceedings of SIGGRAPH'89, pp. 215-222, 1989.
29. Wicke M, Steinemann D, and Gross M. Efficient animation of point-sampled thin shells. Computer Graphics Forum, 24(3):667-676, 2005.
30. Xie H, Wang J, Hua J, Qin H, and Kaufman A. Piecewise C1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. Proceedings of IEEE Visualization'03, pp. 91-98, 2003.
31. Zhou K, Huang X, Xu W, Guo B, and Shum H Y. Direct manipulation of subdivision surfaces on GPUs. ACM Transactions on Graphics, 26(3): 91, 2007.
32. Zwicker M, Pauly M, Knoll O, and Gross M. Pointshop3D: an interactive system for point-based surface editing. ACM Transactions on Graphics, 21(3): 322-329, 2002.