Real-time Meshless Deformation

Xiaohu Guo Hong Qin
State University of New York at Stony Brook
email: {xguo, qin}@cs.sunysb.edu
http://www.cs.sunysb.edu/{∼xguo, ∼qin}

Abstract

In this paper, we articulate a meshless computational paradigm for the effective modeling, accurate physical simulation, and real-time animation of point-sampled solid objects. Both the interior and the boundary geometry of our volumetric object representation only consist of points, further extending the powerful and popular method of point-sampled surfaces to the volumetric setting. We build the point-based physical model upon continuum mechanics, which affords to effectively model the dynamic elastic behavior of point-based volumetric objects. When only surface samples are provided, our prototype system first generates both interior volumetric points and a volumetric distance field with octree structure. The physics of these volumetric points in a solid interior are simulated using the Meshless Moving Least Squares (MLS) shape functions. In sharp contrast to the traditional finite element method (FEM), the meshless property of our new technique expedites the accurate representation and precise simulation of the underlying discrete model, without the need of domain meshing. In order to achieve real-time simulations, we utilize the warped modal analysis method that is locally linear in nature but globally warped to account for rotational deformation. The structural simplicity and real-time performance of our meshless simulation framework are ideal for interactive animation and game/movie production.

Keywords: physics-based animation, point-based animation, meshless method

1 Introduction

The rapid technology advancement of scanning devices has already made large-scale sampled surfaces prevalent and popular in the digital processing pipeline. Developing new algorithms and techniques for point-centered digital processing has become a common and long-term mission in the point-based graphics field. Examining the rich literature of point-centered geometric processing and graphics rendering, the topics of real-time (or interactive) manipulation and animation of point-sampled geometry appear to be less explored. Traditionally in computer animation, meshes are inevitable for physical simulations since the finite element method (FEM) requires an explicit mesh structure. For point-sampled geometry, however, only local vicinity information is needed. Meshes require a much stronger condition that their connectivity should not overlap. The regularities of the mesh, which are essentially determined from its connectivity, are crucial for later-on animation process and numerical computation. In this paper, our goal is to simplify and streamline the entire physical simulation and animation process without any data conversion to meshes in order to further improve its utility and broaden its access by graphics users.

We present a real-time meshless simulation and animation paradigm for point-based volumetric objects. Our system takes any point sampled surfaces as input, generating a volumetric distance field sampled at the center of octree cells for point-sampled surface geometry. The octree decomposition also implicitly defines the geometry of the solid object enclosed by surface points on its boundary. The surface distance field can be utilized to facilitate the embedding of the volumetric point samples. Besides point geometry (on the boundary and at the interior), our physical model is based on continuum mechanics, which enables our system to simulate the dynamic elastic behavior of the point-based scanned objects. Using continuum mechanics, the simulation parameters can be obtained from the technical specification of real materials documented in the typical scientific references, avoiding the tedious parameter fine-tuning and ad-hoc parameter selection as in the case of mass-spring systems (commonly-used in computer animation). The physics in our system are simulated on the volumetric points using the Moving Least Squares (MLS) shape functions, one of the most popular meshless methods that have been developed extensively in mechanical engineering and material science. The fast convergence, ease of adaptive refinement, flexible adjustment of the consistency order and the continuity of derivatives up to any desirable order are some key features of the meshless methods. Note that, the volumetric points employed in our dynamic simulation can be easily generated based on the octree structure outlined above, which necessarily permits the powerful adaptive modeling capability through the local subdivision of any regions of interest in a hierarchical fashion. Traditionally in mechanical engineering, the high computational load associated with most meshless methods is one severe drawback which plagues its further application in computer graphics and animation. In order to drastically speed up the simulation process, we utilize the *Modal Warping* technique [1], in which linear modal analysis is employed to simulate the elastic model in each local reference frame of the simulation node, and the system equations can be globally warped to account for rotational deformation. Through our extensive experiments, we demonstrate that modeling and simulating point-sampled scanned geometry based on the meshless method can streamline the entire digital animation process, and the integration of meshless method with modal analysis is both natural and necessary to significantly improve the animation performance towards real-time simulation of large-scale models using desktop computers.

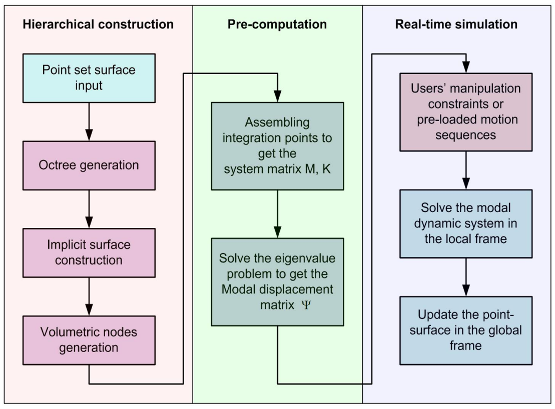
Hierarchical construction
Pre-computation
Real-time simulation
Point set surface input
Octree generation
Implicit surface construction
Volumetric nodes generation
Assembling integration points to get the system matrix M, K
Solve the eigenvalue problem to get the Modal displacement matrix Ψ
Users' manipulation constraints or pre-loaded motion sequences
Solve the modal dynamic system in the local frame
Update the point-surface in the global frame

Figure 1: The general framework of our real-time meshless simulation system.

2 Related work

In this section we will briefly review some related work on physically-based animation, modal analysis, point-based geometry, and meshless methods.

2.1 Physically-based animation and modal analysis

Pioneering work in the field of physically-based animation was carried out by Terzopoulos and his co-workers in [2]. Later, a large number of mesh based methods for both off-line and interactive simulation of deformable objects have been proposed in the field of computer graphics based on either the boundary element method [3] or the finite element method [4]. In 1994, Belytschko et al. proposed the Element Free Galerkin (EFG) method [5] to solve linear elastic problems, specifically the fracture and crack growth problems, in which the Moving Least Squares (MLS) approximation was employed in a Galerkin procedure. However, later-on research on meshless methods are mainly restricted to mechanical analysis field, because the high computational load makes it impractical for computer animations.

Modal analysis is a well established mathematical technique, which was introduced to the graphics field in 1989 by Pentland and Williams [6] as a fast method for approximating deformation. James and Pai [7] implemented a real-time simulation of skin and soft tissues attached to moving rigid bodies by computing modal deformation on graphics hardware. Hauser et al. [8] addressed the manipulation constraints and combined modal analysis with rigid body simulation. Most recently, Choi and Ko [1] proposed a technique called *Modal Warping* that eliminates the linearization artifacts while retaining the efficiency of modal analysis.

2.2 Point-based geometry and meshless methods

Research on point-based geometry has received much attention in the modeling and visualization community in recent years, following Levoy and Whitted's pioneering report [9]. Zwicker et al. presented a system called Pointshop 3D [10] for interactive shape and appearance editing of 3D point-sampled geometry. Alexa et al. [11] used the framework of moving least squares (MLS) projection to approximate a smooth surface defined by a set of points, and they developed several associated resampling techniques to generate an adequate representation of the surface. Amenta and Kil [12] presented a new explicit definition of the point set surfaces in terms of the critical points of an energy function on lines determined by a vector field. Guo et al. [13] equiped point surfaces with level set technique, which can provide various local and global surface editing operations. Later they embedded point clouds into dynamic volumetric implicit models to afford intuitive haptic interaction [14].

Müller et al. [15] presented a method for modeling and animating elastic, plastic, and melting volumetric objects based on the MLS approximation of the gradient of the displacement vector field. In their implementation, point collocation method was used to achieve computational efficiency by avoiding numerical integrals at the expense of the loss of simulation accuracy. Guo and Qin [16] applied the meshless method for simple crack propagation simulation using the level set approach. Pauly et al. [17] combined the meshless method with a highly dynamic surface and volume sampling method that affords complex fracture patterns of interacting and branching cracks. Bao et al. [18] applied the dynamic meshless simulation method to find a physically meaningful transition path between two homeomorphic point-sampled surfaces.

3 Meshless methods

Most recently, meshless (mesh-free) methods [19] have been developed in the field of mechanical engineering to enable solving partial differential equations (PDEs) numerically, based on a set of scattered nodes without having to recourse to an additional mesh structure (which must be put in place for the finite element methods). The advantages of meshless methods for computer animations are multifold: (1) there is no need to generate a mesh of nodes for simulation – the nodes only need to be scattered within the solid object, which is much easier to handle in principle; (2) 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 (3) data management overhead can be minimized during simulation. There are many variants of the meshless methods. In our simulation we use the Moving Least Squares (MLS) shape functions, which has been employed in the Element Free Galerkin (EFG) method introduced in [5], mainly because it has been well-developed with mature techniques, and it has shown a superior rate of convergence and high efficiency. Other variants of available meshless methods can also be adopted into our prototype system in a straightforward way without theoretical obstacles. To make this paper self-contained, we present a brief introduction of the MLS approximation for the definition of shape functions in the following subsection. More detailed derivation can be found in [16].

3.1 MLS shape functions

The shape functions in the EFG method are constructed by using the Moving Least Squares (MLS) approximation 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.

Each node I is associated with a positive weight function w_I of compact support in the analysis domain. From the support of the weight function w_I, we can define the domain of influence of the node: $\Omega_I = \{\mathbf{x} \in R^3 : w_I(\mathbf{x}) = w(\mathbf{x}, \mathbf{x}_I) > 0\}$, 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* $\mathcal{A}(\mathbf{x})$.

If we consider a field function $f(\mathbf{x})$ defined in the analysis domain Ω, we can construct its MLS approximation $\hat{f}(\mathbf{x})$ as:

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

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: $\mathbf{p}^T_{(m=4)} = \{1, x, y, z\}$ in the interest of time performance. The coefficients $\mathbf{a}(\mathbf{x})$ can be derived by minimizing a weighted L_2 norm:

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

where f_I is the nodal field value associated with the node I. Then we can derive the shape function $\phi_I(\mathbf{x})$. 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:

$$f(\mathbf{x}, t) \approx \hat{f}(\mathbf{x}, t) = \sum_{I \in \mathcal{A}(\mathbf{x})} \phi_I(\mathbf{x}) f_I(t), \quad (3)$$

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. Note that the FEM equivalents can also be reached if the weight functions are defined as piecewise-constant entities over each influence domain.

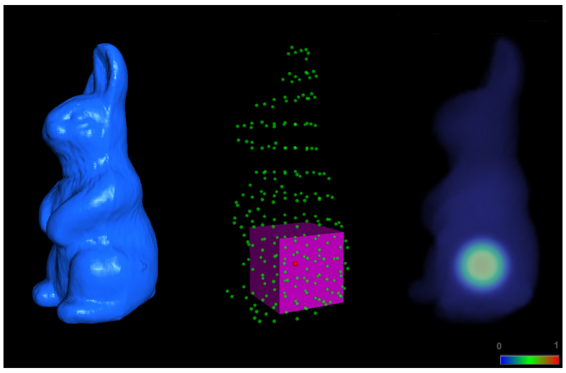

Figure 2: The simulation nodes (green balls), one of the support domain (the pink box), and the volume rendered shape function values.

4 Computational techniques

Our system takes any point set surface as input and utilizes the octree-based hierarchical discretization method for constructing the implicit surface, generating volumetric nodes, and assigning the integration points in order to assemble the system matrices.

4.1 *Hierarchical discretization for meshless dynamics*

The fundamental idea of general 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. First, a discretization without structure does not allow determination of the patches that contribute to a certain integration point without performing an expensive global search. Second, the moment matrix in moving least squares shape function may become invertible if the patch covering conditions (i.e. $\forall \mathbf{x} \in \Omega \quad card\{I : \mathbf{x} \in \Omega_I\} > m$, see [16] for more details) are not satisfied. Last, the effective handling of the interaction between scattered nodes with the geometric boundary (the surface of point clouds in our prototype system) becomes very difficult. From a pure implementation 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.

4.1.1 Octree-based distance field for surface geometry

In our prototype system, the input data is an unstructured point cloud comprising a closed manifold surface. If we conduct our later-on processing and simulation solely on surface points, many difficulties arise. For example, performing inside/outside tests based entirely on surface point information is a forbidding task with many ambiguities. 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 aid in the selection of volumetric points at the interior of solid objects for the dynamic simulation. In our implementation, we utilize *multi-level partition of unity* (MPU) implicit surface construction method proposed by Ohtake et al. [20]. 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. Figure 3 (left) shows the visualization of distance fields using color contours on 2D slices.

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. 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 meshless patches and integration cells.

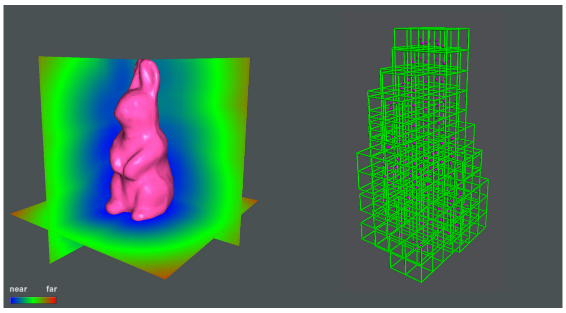

Figure 3: Distance field visualization, and volumetric nodes (pink balls) generated based on octree cells (green lines).

4.1.2 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 meshless 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 3 (right) shows the volumetric nodes generated based on octree-discretization. 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.

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 4). 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

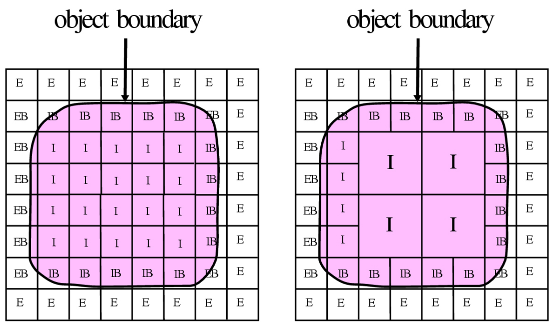

Figure 4: The definition of I, E, IB, and EB octants. We can use octree cells of the same level to place nodes (left); or hierarchically select octree cells (right).

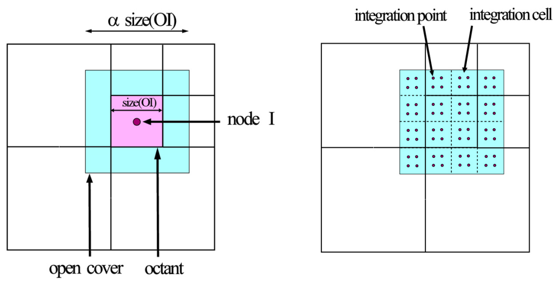

Figure 5: The definition of open cover $\{\Omega_I\}$ regions based on the octree structure (left); the interaction between open covers and integration cells (right).

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 meshless 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 (support region) associated with node i is a cube of size $\alpha \cdot size(O_i)$ centered around node i (see Figure 5 (left)). Both the volumetric nodes and their open cover regions are necessary constituents for meshless dynamics.

The open cover construction based on terminal octants can provide the structure needed to perform efficient neighboring search and patch intersection test. 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 adaptive refinement of the discretization in areas where simulation accuracy is of user's prime interest. Figure 4 shows two different octree discretizations for the same object.

4.1.3 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 5 (right)). 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 Modal analysis for meshless dynamics

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 $\phi_I(\mathbf{x})$ in similar formulae as Equation (3). The partial derivatives with respect to the referencing coordinates x_k can be obtained simply as:

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

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 [6, 7, 8, 1].

4.2.1 Basics of modal analysis

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

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{F}(t) \quad (4)$$

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 $\mathbf{K}\mathbf{x} = \lambda\mathbf{M}\mathbf{x}$, and $\mathbf{\Lambda}$ be the diagonal matrix of eigenvalues, then equation (4) can be transformed to:

$$\ddot{\mathbf{z}} + (\alpha\mathbf{I} + \beta\mathbf{\Lambda})\dot{\mathbf{z}} + \mathbf{\Lambda}\mathbf{z} = \mathbf{\Psi}^T\mathbf{F}, \quad (5)$$

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. The decoupled ODEs in Equation (5) 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.

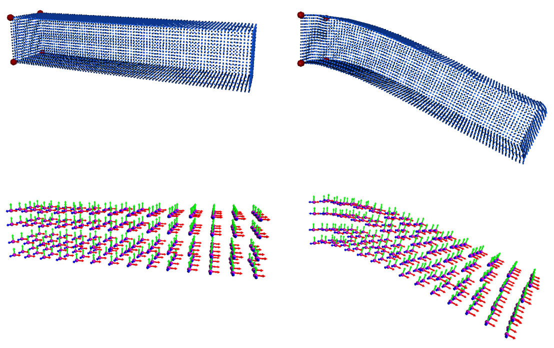
Figure 6: Top: a solid bar with surface point samples before and after bending under gravity (the red balls are fixed positional constraints); Bottom: the local coordinate frame associated with each volumetric simulation node.

4.2.2 Modal warping for rotational deformation

Our Meshless Modal Analysis framework is build upon the *Modal Warping* technique proposed by Choi and Ko [1]. 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 [1].

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

$$\omega = \frac{1}{2}(\nabla \times \mathbf{u})\times = \mathbf{w}\times, \quad (6)$$

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 $\mathbf{z}$:

$$\mathbf{w}(\mathbf{x}) = \frac{1}{2}(\nabla\times)\mathbf{\Phi}(\mathbf{x})\mathbf{\Psi}\mathbf{z} \quad (7)$$

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 6). 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 constant. In order to apply the linear Modal Analysis method, it has been shown in [1] that the non-linear *Euler-Lagrangian* equations

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u})\mathbf{u} = \mathbf{F} \tag{8}$$

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

$$\mathbf{M}\ddot{\mathbf{u}}^L + \mathbf{C}\dot{\mathbf{u}}^L + \mathbf{K}\mathbf{u}^L = \mathbf{R}^T\mathbf{F} \tag{9}$$

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 [1] 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 (9) for $\mathbf{u}^L$ can be reduced to a set of decoupled ODEs:

$$\ddot{\mathbf{z}}^L + \mathbf{C}_z\dot{\mathbf{z}}^L + \mathbf{K}_z\mathbf{z}^L = \mathbf{\Psi}^T(\mathbf{R}^T\mathbf{F}). \tag{10}$$

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 [21] 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

$$\mathbf{A}_z\Delta\dot{\mathbf{z}} = \mathbf{b}_z \tag{11}$$

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 (11) to be solved efficiently.

4.2.3 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:

$$\mathbf{C} = \mathbf{A}_c\mathbf{z} - \mathbf{b}_c = 0 \tag{12}$$

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 (12) can be integrated into the system equation (11) 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 [22]. So we can obtain the constrained equations of motion:

$$\begin{bmatrix} \mathbf{A}_z & \mathbf{A}_c^T \\ \mathbf{A}_c & 0 \end{bmatrix} \begin{bmatrix} \Delta\dot{\mathbf{z}} \\ \lambda h \end{bmatrix} = \begin{bmatrix} \mathbf{b}_z \\ h(-2\eta\dot{\mathbf{C}} - \gamma^2\mathbf{C}) \end{bmatrix}. \tag{13}$$

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 (13) could be solved in real-time.

5 Experimental results

The simulation and rendering parts of our system are implemented on a Microsoft Windows XP PC with dual Intel Xeon 2.8GHz CPUs, 2.0GB RAM, and an nVidia GeForce Fx 5900 Ultra GPU. We have conducted extensive experiments on various scanned point-surface data sets. Table 1 shows the statistics of various models and the corresponding simulation time for each frame. For most of the data sets, the MLS pre-computation for the system matrices takes less than 10 minutes, while the modal decomposition takes less than 1 minute.

model	points	nodes	modes	time
bar	5,634	1,008	64	0.013 s
Igea	134,345	822	64	0.023 s
balljoint	137,062	357	64	0.026 s
rabbit	67,038	1,251	64	0.019 s
Santa	75,781	1,150	128	0.028 s

Table 1: The statistics of various models and their simulation time for each frame.

Figure 7 shows an example of the facial deformation of the Igea model. The facial expression of Igea can be changed by manipulating several positional constraints. The first and second rows of Figure 8 show the deformation of the balljoint and rabbit models under users' manipulation with the bottom of the models fixed. In the bottom row of Figure 8, the Santa Claus model can be simulated based on either users' manipulation, or an input skeletal motion sequence. Please refer to the accompanying video for more simulation details.

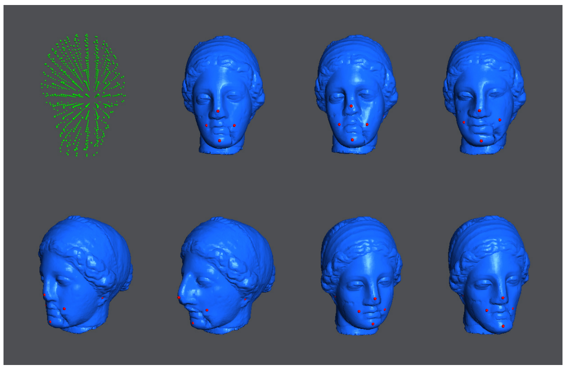
Figure 7: Real-time facial deformation of the Igea model using several positional constraints. The meshless simulation nodes are shown in the top left figure.

6 Conclusion

We have presented a real-time meshless simulation and animation paradigm for volumetric objects, whose interior and surface representations only comprise point samples. The meshless property of our new technique expedites the accurate representation and precise simulation of the underlying discrete model, without the strong need of domain meshing. The meshless dynamics have many unique features, including fast convergence, ease of adaptive refinement, flexible adjustment of the consistency order and the continuity requirement, etc. We exploit the methodology of *Modal Analysis* and adapt the *Modal Warping* technique into our meshless simulation framework to achieve real-time manipulation and deformation. Based on our extensive experiments, we believe that our new paradigm can significantly advance the current state of the knowledge in point-based solid modeling and animation of physical objects. In the near future, the meshless methods and their engineering principles, augmented by novel computational techniques, are expected to open up new research directions in computer graphics, modeling, simulation, and visualization.

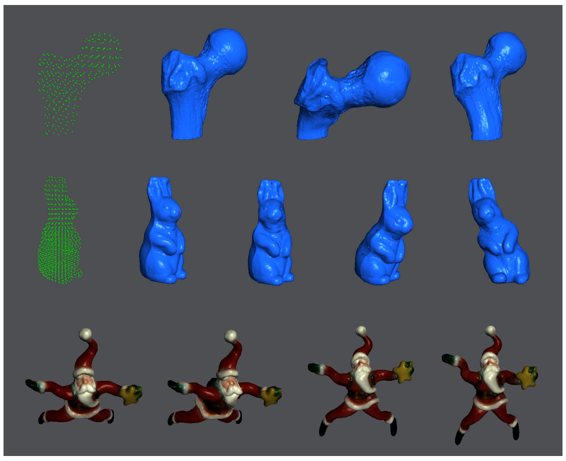
Figure 8: Top: dragging the balljoint while fixing its bottom; Middle: dragging/twisting the rabbit's head with its bottom fixed; Bottom: real-time manipulation of the Santa Claus model.

Acknowledgements

This research work was partially supported by the NSF grant ACI-0328930, the ITR grant IIS-0326388, and Alfred P. Sloan Fellowship to Hong Qin. The Igea, rabbit, balljoint, and santa models are courtesy of Cyberware Inc.

References

[1] M. G. Choi and H. S. Ko. Modal warping: real-time simulation of large rotational deformation and manipulation. *IEEE Trans. Vis. Comput. Graph.*, 1(1):91–101, 2005.

[2] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH*, pages 205–214, 1987.

[3] D. L. James and D. K. Pai. Artdefo: accurate real time deformable objects. In *SIGGRAPH*, pages 65–72, 1999.

[4] G. Debunne, M. Desbrun, M. P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH*, pages 31–36, 2001.

[5] T. Belytschko, Y. Y. Lu, and L. Gu. Element free galerkin methods. *Int. J. Numer. Methods Eng.*, 37:229–256, 1994.

[6] A. Pentland and J. Williams. Good vibrations: model dynamics for graphics and animation. In *SIGGRAPH*, pages 215–222, 1989.

[7] D. L. James and D. K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *SIGGRAPH*, pages 582–585, 2002.

[8] K. Hauser, C. Shen, and J. O'Brien. Interactive deformation using modal analysis with constraints. In *Graphics Interface*, pages 247–255, 2003.

[9] M. Levoy and T. Whitted. The use of points as a display primitive. *Tech. Report85-022, University of North Carolina at Chapel Hill*, 1985.

[10] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop3d: an interactive system for point-based surface editing. *SIGGRAPH*, pages 322–329, 2002.

[11] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.*, 9(1):3–15, 2003.

[12] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, 2004.

[13] X. Guo, J. Hua, and H. Qin. Scalar-function-driven editing on point set surfaces. *IEEE Comput. Graph. Appl.*, 24(4):43–52, 2004.

[14] X. Guo, J. Hua, and H. Qin. Touch-based haptics for interactive editing on point set surfaces. *IEEE Comput. Graph. Appl.*, 24(6):31–39, 2004.

[15] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point-based animation of elastic, plastic, and melting objects. In *ACM SIGGRAPH/Eurographics symp. Computer Animation*, pages 141–151, 2004.

[16] X. Guo and H. Qin. Point-based dynamic deformation and crack propagation. *Tech. Report, Stony Brook University*, 2004.

[17] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph.*, 24(3):957–964, 2005.

[18] Y. Bao, X. Guo, and H. Qin. Physically-based morphing of point-sampled surfaces. *to appear in Comput. Animat. Virtual Worlds*, 2005.

[19] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: an overview and recent developments. *Comput. Meth. Appl. Mech. Eng.*, 139:3–47, 1996.

[20] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. P. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH*, pages 463–470, 2003.

[21] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH*, pages 43–54, 1998.

[22] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. In *SIGGRAPH*, pages 309–312, 1992.