Point-based Dynamic Deformation and Crack Propagation

Xiaohu Guo*
Center for Visual Computing (CVC)
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400

Hong Qin†
Center for Visual Computing (CVC)
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400

Abstract

In this paper, we articulate a novel meshless computational paradigm for the effective modeling, accurate physical simulation, and rapid visualization of solid objects. The uniqueness of our approach is that both the interior and the boundary of our new volumetric representation are point-based, generalizing the powerful and popular method of point-sampled surfaces. We also build the point-based physical model founded upon continuum mechanics, which allow us to effectively model the dynamic behavior of point-based volumetric objects ranging, from elastic deformation to crack propagation. Our prototype system takes any point-sampled surfaces as input and generates both interior volumetric points and a volumetric distance field with an octree structure, which can be utilized to facilitate the crack surface evolution. The physics of these volumetric points in a solid interior are simulated using the Element-Free Galerkin (EFG) method. In sharp contrast to the traditional finite element method, the meshless property of our new technique expedites the accurate representation and precise simulation of the underlying discrete model, without the need of domain remeshing. Furthermore, we develop the new modeling and animation techniques for point-sampled surfaces to dynamically generate cracked surfaces based on the underlying distance field. The accuracy and continuity advantages of the meshless method also enable the direct visualization of the physical quantities of volumetric objects for mechanical and material analysis.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: point-based geometry, mesh-free method, elastic deformation, crack propagation, physically-based modeling, computer animation

1 Introduction

Point sampled geometry has been gaining much popularity in shape modeling, interactive graphics, and visual computing. One key reason for this new interest in points is that the polygonal complexity of graphical models has dramatically increased in the last decade. In computer animation and physical simulation, complex physical effects, such as large deformations and cracks, pose grand technical challenges in terms of maintaining the topological consistency of the underlying meshes. The overhead of managing, processing, and manipulating very large polygonal-mesh connectivity information has given rise to the technical issue of the future utility of polygons as the fundamental graphics primitive.

For the simulation of complex physical phenomena, efficient and consistent surface and volume representations are necessary to facilitate geometric and topological operations. For instance, in simulations of failure processes, we need to model the propagation of cracks along arbitrary and complex paths. This problem, in particular, can not be easily tackled using conventional mesh-based computational methods such as finite element, finite volume, or finite difference techniques. 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 evolution so that mesh lines remain coincident with the discontinuities throughout the evolution. 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.

To overcome the above difficulties associated with mesh structure in computer animation and simulation, in this paper we present a mesh-free modeling, simulation, and visualization 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. Embedding the volumetric point samples into the surface distance field can greatly facilitate the manipulation of the point set surface, e.g., guiding the surface point insertion where large deformations occur. 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 behavior of the point-based objects ranging from elastic deformations to crack propagations. 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 Element-Free Galerkin (EFG) method, one of the most popular mesh-free 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 mesh-free 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 lo-

*e-mail: xguo@cs.sunysb.edu
†e-mail:qin@cs.sunysb.edu

cal subdivision of any regions of interest in a hierarchical fashion. The meshless character of our approach expedites the description of the evolving discrete model in crack simulations and large deformations, while no remeshing of the domain is required. The crack surfaces are modeled using the level set method. In this paper we demonstrate that modeling and animating point-sampled surfaces (that dynamically adapt to large deformations and crack surfaces) can be made much easier by using an underlying surface distance field. Compared with the popular finite element methods, the accuracy and continuity advantages of the mesh-free method make it less difficult to generate smooth interpolation fields (such as stress and strain fields) without any the need of any post-processing. This feature can facilitate direct and fast visualization of the physical properties defined over any volumetric object for either mechanical analysis or educational purposes.

Contributions: This paper's main contributions to the field of solid and physical modeling are:

1. We generalize the concept of point-sampled surfaces to the point-based solid representation for the volumetric interior by utilizing the octree-based space subdivision, which is much simpler and more natural for solid objects.

2. We simulate the physical behavior of solid objects based on continuum mechanics using the mesh-free method, which offers effective modeling of the dynamic behavior of point-based volumetric objects ranging from elastic deformations to crack propagations.

3. We articulate a set of computational techniques that can speed up our dynamic simulations in real applications, including the octree-based volumetric dicretization and integration, the level set method for crack surface representation and propagation, etc.

4. We develop and deploy both surface and volumetric visualization techniques to allow users to intuitively visualize the physical properties of solid objects during the dynamic simulation process.

The remainder of this paper is organized as follows. Section 2 reviews the relevant background. Section 3 discusses the mesh-free methods. Section 4 details the physics of our dynamic system and presents all the computational elements for dynamic simulation. Section 5 addresses the geometry and visualization issues. Section 6 documents our experimental results. Finally, we conclude the paper in Section 7.

2 Related Work

2.1 Point-based Geometry

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 [Levoy and Whitted 1985]. Rusinkiewicz and Levoy [Rusinkiewicz and Levoy 2000] introduced a technique called QSplat, which uses a hierarchy of spheres of different radii to approximate and display a high-resolution model with level-of-detail (LOD) control. Zwicker et al. [Zwicker et al. 2001] proposed the surface splatting technique, which directly renders opaque and transparent surfaces from point clouds without connectivity. Later, they presented a system called Pointshop 3D [Zwicker et al. 2002] for interactive shape and appearance editing of 3D point-sampled geometry. Alexa et al. [Alexa et al. 2003] 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. Pauly et al. [Pauly et al. 2003] presented a free-form shape modeling framework for point-sampled geometry using the implicit surface definition of the moving least squares approximation. Amenta and Kil [Amenta and Kil 2004] 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. Most recently, Mueller et al. [Mueller et al. 2004] 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, however, they did not seek to use the standard mesh-free methods, so more complicated behaviors such as crack propagation were not addressed. In addition, they avoided numerical integrals in the interest of fast simulation. It may be noted that shape modeling and direct continuous field visualization are also difficult to accomplish without the support of shape functions.

2.2 Implicit Surfaces and Level Sets

Implicit functions are well suited for both scientific visualization and modeling tasks in computer graphics [Blinn 1982]. [Bloomenthal and Wyvill 1990] and [Bloomenthal 1997] used skeleton methods to construct implicit surfaces interactively. Each skeletal element is associated with a locally-defined implicit function. Individual functions are then blended to form an implicit surface using a polynomial weighting function that can be interactively controlled by users. Cani and Desbrun [Desbrun and Cani 1998] employed deformable implicit models for animating soft objects. Frisken et al. [Frisken et al. 2000] proposed the adaptively sampled distance fields (ADFs) as an effective representation of geometry and volume data. Turk et al. [Turk and O'brien 2002] introduced the interpolating implicit surfaces for surface reconstruction and shape transformation based on the concept of the variational principle of scattered data interpolation. Ohtake et al. [Ohtake et al. 2003] presented the multi-level partition of unity implicit surface, which allows users to construct surface models from very large sets of points. Shen et al. [Shen et al. 2004] proposed a method for building interpolating or approximating implicit surfaces from polygonal data using a moving least-squares formulation with constraints integrated over the polygons.

Level-set methods were introduced by Osher and Sethian [Osher and Sethian 1988] by representing the contour as the level-set of a scalar-valued function. Desbrun et al. [Desbrun and Cani 1998] and Breen et al. [Breen and Whitaker 2001] used the variant of this method for shape morphing. Malladi et al. [Malladi et al. 1995] and Whitaker et al. [Whitaker et al. 2001] applied this technique to the problem of medical image segmentation. Whitaker [Whitaker 1998] and Zhao et al. [Zhao et al. 2001] employed the level-set method for 3D reconstruction. More recently, Museth et al. [Museth et al. 2002] and Baerentzen et al. [Baerentzen and Christensen 2002] presented the level-set framework for interactively editing implicit surfaces, where they defined a collection of speed functions that can reproduce a set of surface editing operators.

2.3 Physically-based Animation

Pioneering work in the field of physically-based animation and crack simulation was carried out by Terzopoulos and his co-workers in [Terzopoulos et al. 1987] and [Terzopoulos and Witkin 1988].

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 [James and Pai 1999] or the finite element method [Debunne et al. 2001] [Grinspun et al. 2002]. To the best of our knowledge, most graphics researchers rely on mesh-based methods for crack simulation. The work of [Hirota et al. 1998] [Smith et al. 2001] simply break connections or springs between adjacent elements when the force goes beyond a user-specified threshold value. [Mueller et al. 2001] treated objects as rigid bodies in between two consecutive collisions, and used static finite element analysis techniques when collisions occur. The state of the art in fracture modeling for computer graphics is the work of [O'Brien and Hodgins 1999] [O'Brien et al. 2002], which used a pseudo-principal stress and continuous remeshing. Most recently, [Molino et al. 2004] proposed a virtual node algorithm that allows material to be separated along arbitrary piecewise-linear paths through a mesh.

3 Meshfree Methods

During the last two decades, mesh-free (meshless) methods [Belystchko et al. 1996][Li and Liu 2002] 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 finite element methods). The unique advantages of mesh-free methods are multifold: (1) there is no need to generate a mesh of nodes – they only need to be scattered within the solid object, 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 [Belytschko et al. 1994a] to solve linear elastic problems, specifically the fracture and crack growth problems [Belytschko et al. 1994b], 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 mesh-free 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 mesh-free 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.

3.1 EFG Shape Functions

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 subsections.

3.1.1 Moving Least Squares Approximation

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.) [Lancaster and Salkauskas 1986].

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: $\Omega_I = \{\mathbf{x} \in R^3 : w_I(\mathbf{x}) = w(\mathbf{x},\mathbf{x}_I) > 0\}$, 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 $\hat{\mathbf{x}}$ is only affected by those nodes whose weights are non-zero at $\hat{\mathbf{x}}$. We denote the set of such nodes the *active set* $\mathscr{A}(\hat{\mathbf{x}})$. Figure 1 illustrates the mesh-free computational model with rectangular and circular local influence domains, respectively.

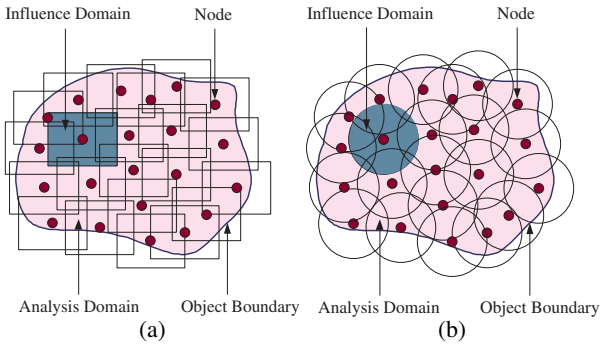


Figure 1: The mesh-free computational model with rectangular (a) and circular (b) support, respectively.

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

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

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

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

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:

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

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 node placement method (which will 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.

3.1.2 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 Computational Techniques

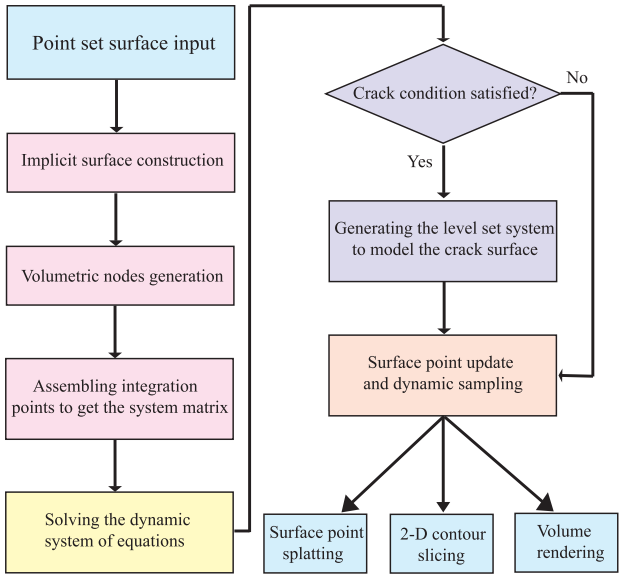


Figure 2: The architecture framework and data pipeline of our meshless modeling, simulation, and visualization system.

The architecture framework and data flow of our meshless modeling, simulation, and visualization system is illustrated in Figure 2. The system takes any point set surface as input and utilizes the octree-based hierarchical discretization method (Section 4.2) for constructing the implicit surface, generating volumetric nodes, and assigning the integration points in order to assemble the system matrices. Then, we can solve the dynamic systems of equations and compute the nodal coefficients associated with each volumetric nodes (Section 4.1). If the stress intensity is greater than the cracking threshold, we model the crack surface propagation using the level set method (Section 4.3). Otherwise, we should update the surface point representation and perform the dynamic sampling when it becomes necessary (Section 5.1). The user can choose different modes of visualization approaches, such as surface point splatting, 2-D contour slicing, and volume rendering, to visualize the dynamic simulation process (Section 5.2).

4.1 Equations of Motion

We shall consider a 3-dimensional body $\mathscr{B}$ which is an open set in the Euclidean space R^3. The body consists of material points X. The material points can be identified with coordinates in a fixed Cartesian system, with basis vectors $e_k, k = 1,2,3$, in a reference domain Ω, i.e., the material point X is identified with the position vector $\mathbf{X} = \sum_k X_k e_k$. The Cartesian coordinate system e_k will be used exclusively, both for the reference and for the current configurations. The motion of the body is described by the mapping $\mathbf{M}$, $\mathbf{x} = \mathbf{M}(\mathbf{X},t)$, where $\mathbf{x}$ is the coordinate of the material point X in the current configuration, $\mathbf{x} = \sum_k x_k e_k$.

In our mesh-free approximation, the motion parameters of the material point X, i.e., the displacements $\mathbf{u} = \mathbf{x} - \mathbf{X}$, velocity $\mathbf{v}$, and acceleration $\mathbf{a}$, can be approximated by using the moving least squares shape functions $\phi_I(\mathbf{X})$ as:

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

$$\dot{\mathbf{u}}(\mathbf{X},t) = \sum_I \phi_I(\mathbf{X})\dot{\mathbf{u}}_I(t), \tag{15}$$

$$\ddot{\mathbf{u}}(\mathbf{X},t) = \sum_I \phi_I(\mathbf{X})\ddot{\mathbf{u}}_I(t). \tag{16}$$

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:

$$\mathbf{x}_{,k}(\mathbf{X},t) = \sum_I \phi_{I,k}(\mathbf{X})\mathbf{x}_I(t). \tag{17}$$

We use the *Euler-Lagrange* equations for our elastic deformation:

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{u}})}{\partial \dot{\mathbf{u}}}\right) + \mu\dot{\mathbf{u}} + \frac{\partial V(\mathbf{u})}{\partial \mathbf{u}} = \mathbf{F}_{ext}, \tag{18}$$

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:

$$T = \frac{1}{2}\int_\Omega \rho(\mathbf{x})\dot{\mathbf{u}}\cdot\dot{\mathbf{u}}d\Omega = \frac{1}{2}\sum_{I,J} M^{IJ}\dot{\mathbf{u}}_I\cdot\dot{\mathbf{u}}_J, \tag{19}$$

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:

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{u}})}{\partial \dot{\mathbf{u}}}\right) = \mathbf{M}\mathbf{a}, \tag{20}$$

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:

$$\varepsilon_{ij} = \frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \delta_{kl}\frac{\partial u_k}{\partial X_i}\frac{\partial u_l}{\partial X_j}. \tag{21}$$

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

$$\tau_{ij} = 2G\{\frac{\nu}{1-2\nu}tr(\varepsilon)\delta_{ij} + \varepsilon_{ij}\}, \tag{22}$$

where $tr(\varepsilon) = \sum_{ij}\delta_{ij}\varepsilon_{ij}$. 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:

$$V = G\int_\Omega \{\frac{\nu}{1-2\nu}tr^2(\varepsilon) + \sum_{ijkl}\delta_{ij}\delta_{kl}\varepsilon_{ik}\varepsilon_{jl}\}d\Omega, \tag{23}$$

By combining the above Equations (14), (21), 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.

4.1.1 Solving the System

The quadratic strain (Equation (21)) makes the system Equations (18) a nonlinear system, in which the internal elastic force is not a linear combination of the nodal displacements. To make the equations easier to solve, we make simplifications by linearizing the equations of motion at the beginning of each time step as in [Baraff and Witkin 1998]. By applying their implicit solver to our formulation, the resulting equations become:

$$\Delta\mathbf{u} = \Delta t(\dot{\mathbf{u}} + \Delta\mathbf{v}), \tag{24}$$

$$(\mathbf{M} + \Delta t\mu\mathbf{I} + (\Delta t)^2\mathbf{S})\Delta\mathbf{v} = \Delta t(\mathbf{F}_{ext} - \frac{\partial V}{\partial \mathbf{u}} - \mu\dot{\mathbf{u}} - \Delta t\mathbf{S}\dot{\mathbf{u}}), \tag{25}$$

where Δt is the time step, $\Delta\mathbf{u}$ is the change in $\mathbf{u}$ during the time step, $\Delta\mathbf{v}$ is the change in the velocity $\dot{\mathbf{u}}$ during the time step, $\mathbf{M}$ is the mass matrix, $\mathbf{I}$ is the identity matrix, and $\mathbf{S}$ is the stiffness matrix. We can solve Equation (25) using a Conjugate Gradient (CG) solver, and then substitute $\Delta\mathbf{v}$ into Equation (24) to obtain $\Delta\mathbf{u}$.

4.1.2 Position Constraints

In order for our objects to interact with other objects, position constraints are important and must be enforced. In general, the MLS shape functions lack the Kronecker delta function property and result in $\mathbf{u}(\mathbf{X}_I) \neq \mathbf{u}_I$. Therefore, we would face difficulties when imposing essential boundary conditions on the boundaries of the analysis domain. The approach of Capell et al. [Capell et al. 2002] provides us a solution to deal with position constraints. We can formulate the position constraints in the form:

$$\mathbf{d}_c(t) = \sum_I \phi_I(\mathbf{X}_c)\mathbf{u}_I, \tag{26}$$

where $\mathbf{X}_c$ is the constrained position of the object, and $\mathbf{d}_c(t)$ is the displacement at $\mathbf{X}_c$, which is known a priori. According to Equation (24), we have:

$$\sum_I \phi_I(\mathbf{X}_c)\Delta\mathbf{v}_I = \frac{\mathbf{d}_c(t+\Delta t) - \mathbf{d}_c(t)}{\Delta t} - \sum_I \phi_I(\mathbf{x}_c)\dot{u}_I. \tag{27}$$

Note that the r.h.s. of this equation is a constant. By solving this equation using singular value decomposition (SVD) at the beginning of each time step, we can get the constrained components of $\Delta\mathbf{v}$. The modified CG solver of Baraff and Witkin in [Baraff and Witkin 1998] can be directly employed by projecting and filtering out certain components of $\Delta\mathbf{v}$ corresponding to the constrained nodes (please refer to [Baraff and Witkin 1998] for details). Figure 3 shows an example of the elastic deformation of a straight bar with left part fixed. The stress intensity distribution is visualized using the volume rendering technique discussed in section 5.2.

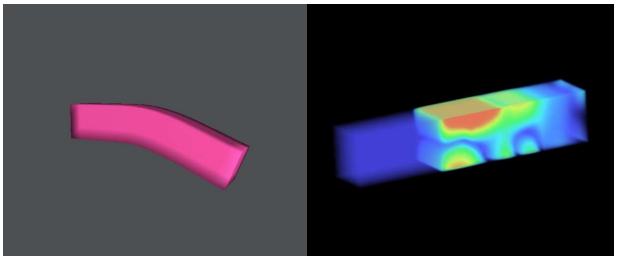


Figure 3: (a) The elastic deformation of a straight bar; (b) the stress intensity distribution inside the object visualized by volume rendering.

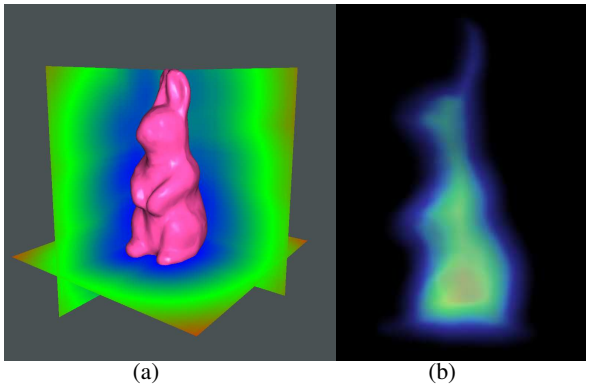
4.2 Hierarchical Discretization for Mesh-free Dynamics

The fundamental idea of general mesh-free 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 mesh-free 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 $\mathbf{A}$ in moving least squares shape function may become invertible if the patch covering conditions (e.g., Equation (8)) 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.

4.2.1 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 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. But most of all, conducting the dynamic simulation only on the solid boundary is far less physically meaningful, leading to incorrect simulation results. 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. [Pauly et al. 2003] 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 pro-

jection operator [Xie et al. 2003]. Guo et al. [Guo et al. 2004b] 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 4 shows the visualization of distance fields using color contours on 2D slices and volume rendering techniques (see section 5.2 for details on our visualization techniques).



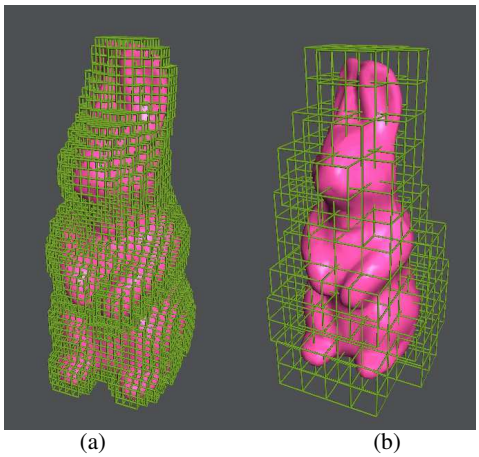
(a) (b)

Figure 4: Distance field visualization for point set surfaces: (a) contours on 2D slices; (b) volume rendering.

In our implementation, we utilize *multi-level partition of unity*(MPU) implicit surface construction method proposed by Ohtake et al. [Ohtake et al. 2003]. 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 [Klaas and Shephard 2000]. 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.

4.2.2 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 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 5 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.



(a) (b)

Figure 5: Octree-based discretization for surface distance field construction (a) and volumetric node placement (b). The size of the terminal octants in (b) is much larger than that of (a).

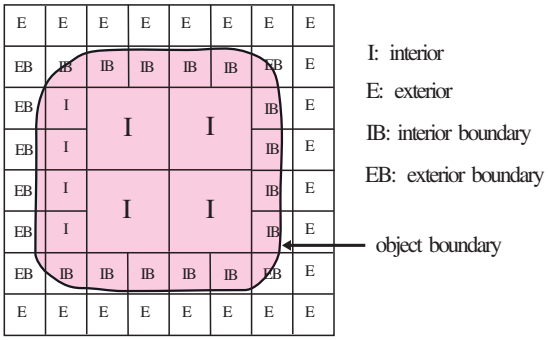


Figure 6: The definition of interior, exterior, interior boundary, and exterior boundary octants for mesh-free simulation.

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 6). 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 7). Both the volumetric nodes and their open cover regions are necessary constituents for mesh-free dynamics.

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 [Klaas and Shephard 2000] that by choosing a suitable size for α, the validity of the open

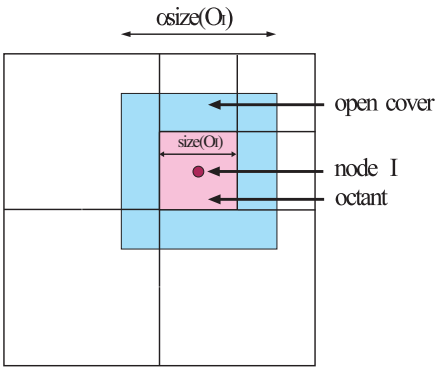


Figure 7: The definition of open cover $\{\Omega_I\}$ regions based on the octree structure for mesh-free patches.

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).

4.2.3 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 8). 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.

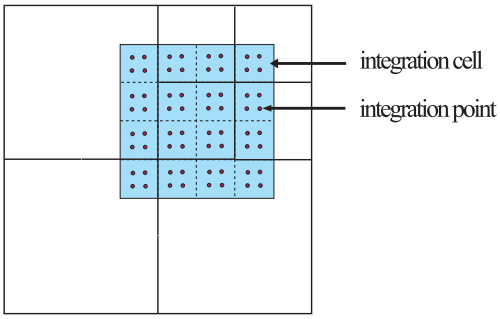


Figure 8: The interaction between open covers and integration cells. Integration points for Gaussian quadrature are also shown.

4.3 Crack Propagation

The main computational techniques currently used in fracture mechanics are the finite element method, finite difference method, and the boundary integral method. One of the most difficult aspects of modeling the evolution of cracks is the need to tightly couple an evolving solid model representation of the body with the discretization for each stage of the propagation. The ability of mesh-free methods to minimize or simplify changes to the discrete model is why they are promising alternatives to the traditional mesh-based approaches. Typically, there are two aspects of crack propagations that are of interest: the physical model undergoing the crack evolution and the representation of the evolving geometry. We use the simplified Rankine condition [Rankine 1872] of maximal principle stress to decide both whether and how the material cracks. If the maximum eigenvalue of τ exceeds a threshold, a crack (with cracking speed $\mathbf{v}_c$ proportional to the maximum eigenvalue of τ) should be generated. Secondary fractures on the cracking surface can be given higher thresholds to help reduce spurious branching in practice.

4.3.1 Discontinuity of the Shape Functions

Mesh-free methods produce arbitrarily smooth shape functions, which is undesirable in cases of discontinuities, such as cracks. When a crack is generated in a body, the dependent variables (e.g., the displacements), 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. The simplest way to introduce discontinuities into mesh-free approximations is to use the visibility criterion in their construction [Belytschko et al. 1994b] [Krysl and Belytschko 1999]. In this method, the boundaries of the body and any interior surfaces of discontinuity are considered opaque when constructing the weight functions, i.e., the line from a point to a node is imagined to be a ray of light. If the ray encounters an opaque surface, such as the boundary of a body or an interior discontinuity, it is terminated, and the point is not included in the domain of influence. In the following subsection, we will show that the visibility criterion can be easily implemented when the cracking surface is modeled as a scalar field.

4.3.2 Crack Surface Modeling based on Level Sets

A key feature of crack growth simulation is the evolving geometry (crack surface). The growth of the crack changes the geometric model, and implementing these changes is one of the most difficult tasks of crack propagation simulations, particularly in 3D. The growth of a crack in 3D can perhaps be better viewed as the evolution of a surface by the motion of a curve (crack front). However, the path of the curve must be explicitly remembered (i.e., stored), as it constitutes the crack surface. Thus, an ideal method for our prototype system would provide the evolution of the curve and can be coupled with our point-based surface representation simultanuously.

Burchard et al. [Burchard et al. 2001] presented numerical simulations of a level set based method for moving curves in 3D. The level set method is a general tool for the description of evolving surfaces, and has been applied extensively in image processing, computer vision, scientific visualization, and shape modeling, where it has been extremely successful, especially when topological changes in the interface, i.e., merging and breaking, occur. In [Stolarska et al. 2001] the level set methods were first applied to

crack problems, where two orthogonal level sets were used, one for the crack surface, the second to locate the crack tip. Recently, Guo et al. [Guo et al. 2004a] [Guo et al. 2004b] proposed coupling the point set surfaces with level sets to facilitate surface modeling and editing. We have observed that coupling implicit surfaces with explicit point-based representation can provide us with a much more powerful scheme that takes advantage of both representations, such as topology change handling, efficient rendering, etc. And it can also greatly facilitate the visibility test (section 4.3.1) between the integration points and the mesh-free nodes. In this paper, we continue to make use of the dual representations for the purpose of modeling the crack surface.

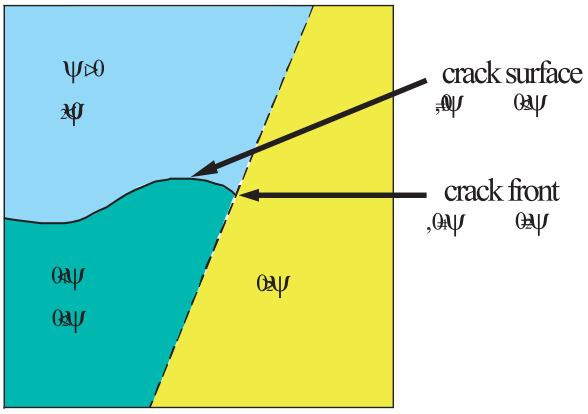


Figure 9: Construction of the two level set functions for the crack surface (ψ_1) and crack front (ψ_2).

The level set method is a numerical technique for tracking the motion of interfaces. The interface of interest is represented as the zero level set of an implicit function $\psi(\mathbf{x}(t),t)$. This function is one dimension higher than the dimension of the interface. We model the crack by two orthogonal level sets (see Figure 9):

1. The ψ_1 level set, called crack surface level set; its zero iso-surface corresponds to the crack surface.
2. The ψ_2 level set, called front level set; the intersection of the crack surface zero level set ($\psi_1 = 0$) with the front zero level set ($\psi_2 = 0$) gives the crack front.

$\psi_1(\mathbf{x},t)$ and $\psi_2(\mathbf{x},t)$ are assumed to be signed distance functions. The crack surface and crack front are given as:

$$\textit{Crack Surface}: \quad \psi_1(\mathbf{x},t) = 0,\ \psi_2(\mathbf{x},t) < 0 \tag{28}$$

$$\textit{Crack Front}: \quad \psi_1(\mathbf{x},t) = 0,\ \psi_2(\mathbf{x},t) = 0 \tag{29}$$

The level sets are only updated in a small sub-domain around the crack surface, which we call the level set sub-domain. For multiple crack surfaces and crack fronts, we can use multiple different level sets to model them. The point sampled crack surface can be generated by projecting the grid points near the crack surface onto the zero level set of the iso-surface [Co et al. 2003]. The crack front is also represented explicitly as a linked list of point samples.

In order to find the position of a point in the level set sub-domain relative to the crack front an additional level set function must be introduced. This function is defined as the perpendicular distance to the crack front, and is denoted as $\varphi(\mathbf{x},t)$. The φ level set can be computed for each grid point in the level set sub-domain using the Fast Marching Method (FMM) [Sethian 1999a], which is based on solving the Eikonal equation $|\nabla\varphi| = 1$ using level sets. Another key ingredient in applying level set methods to crack growth is the extension of the velocity field $\mathbf{v}_c$ (crack speed) from the crack front to the entire level set sub-domain. This can also be performed by an extension of the FMM. For details of the Fast Marching Method, please refer to [Sethian 1999b].

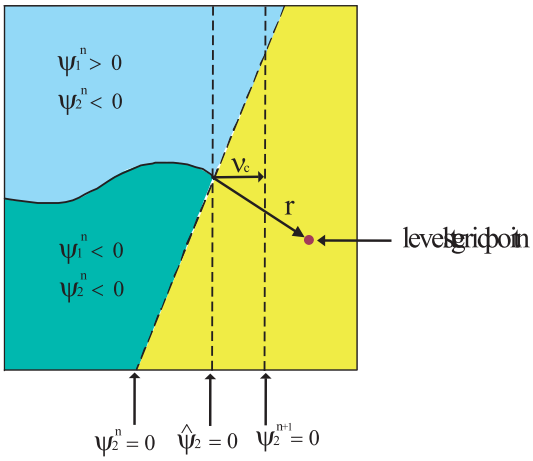


Figure 10: Construction of the crack surface (ψ_1) and crack front (ψ_2) level set functions.

Assume that the values of φ, ψ_1, and ψ_2 at time n are φ^n, ψ_1^n, and ψ_2^n, respectively. The update procedure of level sets in 3D can be described as follows (see Figure 10):

1. Compute the level set function of φ^{n+1} from the crack front using the FMM. Then the position of each grid point relative to the crack front is given by $\mathbf{r} = |\varphi|\nabla\varphi$.
2. Extend the velocity vector $\mathbf{v}_c$ given on the crack front into the entire level set sub-domain by an extension of FMM.
3. Rotate the crack front level set ψ_2^n so that it is orthogonal to the velocity field $\mathbf{v}_c$ obtained from step (2). This is performed geometrically by $\hat{\psi}_2 = \mathbf{r} \cdot \frac{\mathbf{v}_c}{\|\mathbf{v}_c\|}$, where $\hat{\psi}_2$ is the rotated temporary crack front level set.
4. Extend the crack geometrically by computing the crack surface level set $\psi_1^{n+1} = \pm\|\mathbf{r} \times \frac{\mathbf{v}_c}{\|\mathbf{v}_c\|}\|$ in the region where $\hat{\psi}_2 > 0$. The sign of ψ_1^{n+1} is chosen so that it is consistent with the current sign on a given side of the crack.
5. Update the crack front level set $\psi_2^{n+1} = \hat{\psi}_2 - \Delta t\|\mathbf{v}_c\|$ in the region where $\hat{\psi}_2 > 0$.

Figure 11 shows the crack surface propagation inside a straight bar due to the gravity force with its right part fixed.

5 Model Representation and Visualization

5.1 Point-based Surface Geometry

The only input of our prototype system is a point sampled surface. It is maintained during deformations through dynamic sampling. We construct the volumetric distance field from the point cloud using the multi-level partition of unity implicit surface construction method [Ohtake et al. 2003]. For crack surfaces, point samples can be generated by projecting the grid points near the crack surface onto the zero level set of the iso-surface. The implicit surfaces are

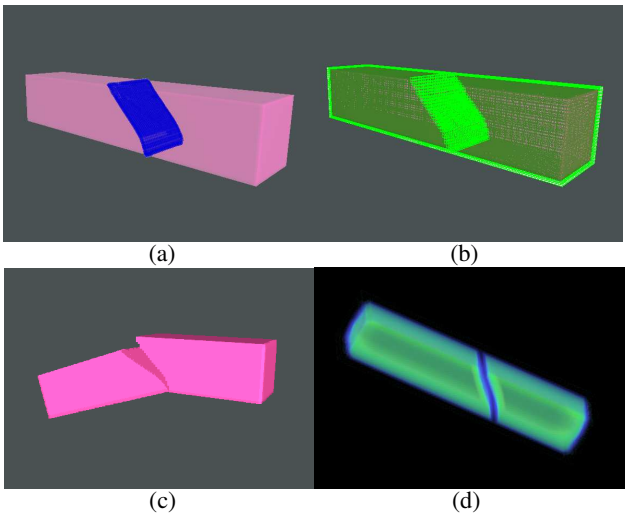


Figure 11: Crack surface propagation inside a bar: (a)the point set representation of the crack surface; (b)level set grids inside the bar; (c)the broken bar; (d)volume rendering of the distance field inside the bar.

maintained in the reference domain of the deformed object during deformations. Large deformations may cause strong distortions in the distribution of sample points on the surface that can lead to an insufficient local sampling density. We have to include new samples where the sampling density becomes too low. To achieve this, we utilize the first fundamental form as in [Pauly et al. 2003] to measure the surface stretch to detect regions of insufficient sampling density. Then, we have to insert new sample points where the local distortion becomes too large. The inserted points can be projected onto the zero level set of the surface distance field to get their location in the reference domain. It might be desirable to eliminate point samples in regions where the surface is squeezed to keep the overall sampling distribution uniform. We implemented an iterative simplification method introduced in [Pauly et al. 2002]. However, the deleted points are just temporarily placed in a "recycle bin". They may be reused when they are re-inserted.

5.2 Visualization Techniques

The physical attribute distribution in a volumetric solid object can be visualized in a number of ways, for example, by color contours on a 2D slice, iso-contour splatting, or by direct volume rendering.

Volume rendering refers to techniques which produce a X-ray-like image directly projected from the volume data via ray integration. It enhances 3D visualization of imaged solid objects by providing translucent rendering. In addition to the standard 3D image analysis tools, volume rendering allows the user to interactively define thresholds for opacity, color application, and brightness. Translucent rendering of volumetric data provides more information about a spatial relationship of different structures than standard 3D surface rendering and 2D slice contouring. Direct volume rendering affords to quickly isolate regions of interest, and quickly provides 3D spatial information for enhanced mechanical analysis, and aids in education.

There are two principle approaches to volume rendering [Westover 1989] [Westover 1990]: backward mapping (ray casting) algorithms that map the image plane onto the data by shooting rays from pixels into the data space, and forward mapping (splatting) algorithms that map the data onto the image plane. If we treat our volumetric nodes as the scattered data used for volume rendering, the shape function can be considered as the reconstruction kernel in volume rendering. For forward mapping, sampling the footprint function for each pixel involves an integration. For our MLS shape functions, it is difficult to integrate analytically, and the renderer must use discrete methods. Unlike the Radial Basis Functions, the MLS shape functions are not rotationally symmetric, which makes it impossible to be pre-computed for all view directions. To perform volume rendering of the physical attributes on our volumetric object, we simply use the integration points constructed in Section 4.2.3 since the attribute values are already available at these points. We then utilize the standard splatting algorithm to render these integration points using spherical gaussian kernels. Figure 12 shows an example of both point-based splatting of the surface and volume rendering of the stress intensity field.

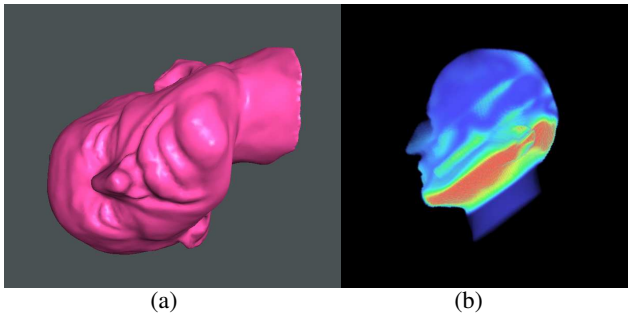


Figure 12: (a) Point-based splatting of the surface of the max plank model undergoing elastic deformations. (b) Volume rendering of the stress intensity field.

6 Implementation and Discussion

The simulation and rendering parts of our system are implemented on a Microsoft Windows XP PC with dual Intel Xeon 2.0GHz CPUs, 1.5GB RAM, and an nVidia GeForce Fx 5200 Ultra GPU. The entire system is written using Microsoft Visual C++, and the graphics rendering component is built upon OpenGL.

Table 1: Simulation speed (sec/frame) for both elastic deformation and crack propagation. $T1$ is the timing for elastic deformation; $T2$ is the timing for crack propagation; $T3$ stands for the level set update timing for crack surface.

model	surfels	nodes	$T1$	$T2$	$T3$
straint bar	6144	64	0.01579	0.27145	0.12425
max plank	15,002	251	0.06787	0.84713	0.25372

Table 1 shows the statistics of the performance of our physical simulation system on several point data sets. The third column shows the timing for elastic simulation, while the fourth column is for physical simulation of cracks. And the fifth column give us the timing for level set update of the crack surfaces. In our experiments, we use the implicit integration scheme with a time step of 0.01 seconds without stability problems. For simulations of elastic deformations, the system matrices can be precomputed. For crack propagations, the system matrices need to be updated to accommodate the changes of the local reference domain of each node. The level set crack surface propagation can be performed efficiently since the level set updates are only computed in the subdomain around the crack surface without any remeshing issues associated with finite

element methods. One limitation of our current framework is that we can only handle volumetric objects, since the nodes' distribution has to satisfy the non-degeneracy condition in order for the moment matrix to be invertible. Extending our volumetric framework to thin shell structure is one of our future work.

7 Conclusion

We have presented a new mesh-free modeling, simulation, and visualization paradigm for volumetric objects, whose interior and surface representations are both point clouds. In particular, our system takes any point sampled surface as input, generates octree-based volumetric points for use in dynamic simulation, and simulates the elastic deformations and cracks using the Element-Free Galerkin method based on continuum mechanics. Besides dynamics, our surface point samples are also used to produce a volumetric distance field that can speedup the geometric queries and manipulations at the same time. The mesh-free dynamics have many unique features, such as fast convergence, ease of adaptive refinement, flexible adjustment of the consistency order and the continuity requirement, etc. The meshless character of our approach expedites the effective modeling of the time-evolving discrete model in crack propagations, while no remeshing of the domain is required. We use the level set method to accurately track the crack surface, which can facilitate modeling and animating point-sampled surfaces that dynamically adapts to crack surfaces. Compared with traditional finite element methods, the high accuracy and continuity characteristics of our approach make it possible to perform rapid visualization of the physical properties of the volumetric solid object without any post-processing operation. Based on our extensive experiments on the mesh-free simulations, we believe that our new paradigm can significantly advance the current state of the knowledge in point based solid modeling and visualization of physical objects. In the near future, the mesh-free methods and their engineering principles are expected to open up new research directions in computer graphics, modeling, simulation, and visualization.

Acknowledgment

This research was supported in part by the NSF grants IIS-0082035 and IIS-0097646, and Alfred P. Sloan Fellowship. The rabbit model is courtesy of Cyberware Inc.

References

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE TVCG 9*, 1 (January-March), 3–15.

AMENTA, N., AND KIL, Y. J. 2004. Defining point-set surfaces. *ACM Trans. Graph. 23*, 3, 264–270.

BAERENTZEN, J. A., AND CHRISTENSEN, N. J. 2002. Volume sculpting using the level-set method. In *International Conference on Shape Modeling and Applications*, 175–182.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *SIGGRAPH*, ACM Press, 43–54.

BELYSTCHKO, T., KRONGAUZ, Y., ORGAN, D., FLEMING, M., AND KRYSL, P. 1996. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering 139*, 3–47.

BELYTSCHKO, T., LU, Y. Y., AND GU, L. 1994. Element free galerkin methods. *International Journal for Numerical Methods in Engineering 37*, 229–256.

BELYTSCHKO, T., LU, Y. Y., AND GU, L. 1994. Fracture and crack growth by element-free galerkin methods. *Modeling Simulations for Materials Science and Engineering 2*, 519–534.

BLINN, J. F. 1982. Generalization of algebraic surface drawing. *ACM Trans. on Graphics 1*, 3, 235–256.

BLOOMENTHAL, J., AND WYVILL, B. 1990. Interactive techniques for implicit modeling. *Computer Graphics 2*, 24 (March), 109–116.

BLOOMENTHAL, J. 1997. *Introduction to implicit surfaces*. Morgan Kaufmann. Edited by J. Bloomenthal with C. Bajaj, J. Blinn, etc.

BREEN, D. E., AND WHITAKER, R. T. 2001. A level-set approach for the metamorphosis of solid models. *IEEE Trans. on Visualization and Computer Graphics 7*, 2, 173–192.

BURCHARD, P., CHENG, L.-T., MERRIMAN, B., AND OSHER, S. 2001. Motion of curves in three spatial dimensions using a level set approach. *Journal of Computational Physics 170*, 720–741.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIC, Z. 2002. Interactive skeleton-driven dynamic deformations. In *SIGGRAPH*, ACM Press, 586–593.

CO, C. S., HAMANN, B., AND JOY, K. I. 2003. Iso-splatting: A point-based alternative to isosurface visualization. In *11th Pacific Conference on Computer Graphics and Applications*, IEEE CS Press, 325–334.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, 31–36.

DESBRUN, M., AND CANI, M.-P. 1998. Active implicit surface for animation. In *Graphics Interface*, 143–150.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH*, 249–254.

GRINSPUN, E., KRYSL, P., AND SCHRODER, P. 2002. Charms: a simple framework for adaptive simulation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 281–290.

GUO, X., HUA, J., AND QIN, H. 2004. Point set surface editing techniques based on level-sets. In *Computer Graphics International*, 52–59.

GUO, X., HUA, J., AND QIN, H. 2004. Scalar-function-driven editing on point set surfaces. *IEEE Computer Graphics and Applications 24*, 4, 43–52.

HIROTA, K., TANOUE, Y., AND KANEKO, T. 1998. Generation of crack patterns with a physical model. 126–137.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 65–72.

KLAAS, O., AND SHEPHARD, M. S. 2000. Automatic generation of octree-based three-dimensional discretizations for partition of unity methods. *Computational Mechanics 25*, 296–304.

KRYSL, P., AND BELYTSCHKO, T. 1999. The element free galerkin method for dynamic propagation of arbitrary 3-d cracks. *International Journal for Numerical Methods in Engineering 44*, 767–800.

LANCASTER, P., AND SALKAUSKAS, K. 1986. *Curve and Surface Fitting: An Introduction*. Academic Press, London.

LEVOY, M., AND WHITTED, T. 1985. The use of points as a display primitive. *Technical Report85-022, University of North Carolina at Chapel Hill*.

LI, S., AND LIU, W.-K. 2002. Meshfree particle methods and their applications. *Applied Mechanics Review 54*, 1–34.

MALLADI, R., SETHIAN, J. A., AND VEMURI, B. C. 1995. Shape modeling with front propagation: A level set approach. In *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 158–175.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3, 385–392.

MUELLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, Springer-Verlag New York, Inc., 113–124.

MUELLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point-based animation of elastic, plastic, and melting objects. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Computer Animation*.

MUSETH, K., BREEN, D. E., WHITAKER, R. T., AND BARR, A. H. 2002. Level set surface editing operators. In *SIGGRAPH '02 Proceedings*, 330–338.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 137–146.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 291–294.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. In *SIGGRAPH*, 463–470.

OSHER, S., AND SETHIAN, J. A. 1988. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics 79* (November), 12–49.

PAULY, M., GROSS, M., AND KOBBELT, L. 2002. Efficient simplification of point-sampled surfaces. *IEEE Visualization*, 163–170.

PAULY, M., KEISER, R., KOBBELT, L., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *SIGGRAPH*, 641–650.

RANKINE, W. 1872. *Applied Mechanics*. Charles Griffen and Company, London.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. Qsplat: A multiresolution point rendering system for large meshes. *SIGGRAPH*, 343–352.

SETHIAN, J. A. 1999. Fast marching methods. *SIAM Review 41*, 2, 199–235.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods*, second ed. Cambridge University Press.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *SIGGRAPH*, ACM Press.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. 81–91.

STOLARSKA, M., CHOPP, D. L., MOES, N., AND BELYTSCHKO, T. 2001. Modelling crack growth by level sets in the extended finite element method. *International Journal for Numerical Methods in Engineering 51*, 943–960.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications 8*, 6, 41–51.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM Press, 205–214.

TURK, G., AND O'BRIEN, J. F. 2002. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21*, 855–873.

WESTOVER, L. 1989. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, ACM Press, 9–16.

WESTOVER, L. 1990. Footprint evaluation for volume rendering. In *SIGGRAPH*, ACM Press, 367–376.

WHITAKER, R., BREEN, D., MUSETH, K., AND SONI, N. 2001. Segmentation of biological volume datasets using a level-set framework. *Volume Graphics*, 249–263.

WHITAKER, R. T. 1998. A level-set approach to 3d reconstruction from range data. In *International Journal of Computer Vision*, 203–231.

XIE, H., WANG, J., HUA, J., QIN, H., AND KAUFMAN, A. 2003. Piecewise c1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. In *Proceedings of the 14th IEEE Visualization*, 91–98.

ZHAO, H.-K., OSHER, S., AND FEDKIW, R. 2001. Fast surface reconstruction using the level set method. In *Proc. 1st IEEE Workshop on Variational and Level Set Methods*, 194–202.

ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. 2001. Surface splatting. *SIGGRAPH*, 371–378.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop3d: An interactive system for point-based surface editing. *SIGGRAPH*, 322–329.