A DISCRETE MECHANICS MODEL FOR DEFORMABLE BODIES

Johan Jansson Joris S. M. Vergeest

Faculty of Design, Engineering and Production
Delft University of Technology
The Netherlands

Abstract

This paper describes the theory and implementation of a discrete mechanics model for deformable bodies, incorporating behavior such as motion, collision, deformation etc. The model is fundamentally based on inter-atomic interaction, and recursively reduces resolution by approximating collections of many high-resolution elements with fewer lower-resolution elements. The model can be viewed as an extended mass-spring model. We begin by examining the domain of conceptual design, and find there is a need for physics based simulation, both for interactive shape modeling and analysis. We then proceed with describing a theoretical base for our model, as well as pragmatic additions. Applications in both interactive physics based shape modeling and analysis are presented. The model is aimed at conceptual mechanical design, rapid prototyping, or similar areas where adherence to physical principles, generality and simplicity are more important than metric correctness.

Key words:
mechanics model, deformation, collision, deformable bodies, geometric modeling, conceptual design, virtual claying

1 Introduction

The context of this paper is conceptual mechanical design. It is well known that the conceptual stage of the design process still is largely unsupported by computer tools. We believe that such support can greatly increase efficiency, by for example providing rapid verification of early design ideas, or support for quick description and modification of non-detailed 3D geometry, i.e. the equivalent of sketching.

There are many possible aspects of such support: natural interaction, vague description, virtual environments, physical simulation, rapid prototyping etc. Our research group, the Integrated Concept Advancement (ICA) group [ICA group web site, 2000],

is researching some of these areas. This paper will focus on physical simulation. We will describe a mechanics model and implementation which can be used for geometric modeling and physical analysis in the conceptual design phase.

For computer tools to actually support, instead of hinder this phase, several requirements must be met. There have been studies determining what these requirements are [Wiegers, et. al., 1999], [Delsinger, et. al. 2000]. The relevant requirements posed for this specific domain are:

(1) Natural interaction methods - interaction with virtual objects should be experienced as interaction with real objects.
(2) Rapid feedback and evaluation - interaction should not be hindered by poor resolution and time lags

Notably missing from the list of requirements is metric accuracy. This is the main difference compared to traditional CAD applications, we can sacrifice metric accuracy for other properties, interactivity for example. However, this does not give as much freedom as might seem. We still need to make sure that the phenomena we are trying to simulate are physically correct, especially if we want to perform physical analysis.

We have developed a mechanics model which is particle system based, and in which accuracy is dependent on the resolution of description. We reason recursively, the model defines lower-resolution elements which approximate a collection higher-resolution elements. This means that we can employ induction as a means of making sure the model is physically correct. If we can show that some ideal resolution is physically correct, and show that the operation of reducing resolution some given step does not remove this property, we can assume that low resolutions also will have this property. With this, we are not aiming for a formal proof of correctness, but want to show the philosophy of the model. Also, this kind of reasoning is very dependent on the definitions of the terms we are using. It is evident that each operation of reducing resolution removes some kind of correctness, we just want a guarantee that we are not removing any physical principles.

Assuming we have such a model which covers deformable bodies, we can consider various applications. The most direct application is analysis. During the early stage of design, it can be desirable to get indications of what kind of physical implications some given design directions will have, essentially a pruning of the design space. Ford Motor Company has expressed such a need in a paper [Ping, Nanxin, 2000]. Apparently there is a need for simulation data at the early design stage of new cars, and traditional simulation is too expensive and time consuming to be applicable. Although the approach described in that paper is of a different type than what we propose (it proposes a Design of Experiment (DoE) approach), it clearly shows a need for such analysis. Even if our approach at this stage might not be suited for the car industry, we can assume that the same need exists in other related industries.

Another perhaps more interesting application is virtual environments. Given the coverage of the model (deformable bodies), and an interface which can present a user as a body in the environment, we can reformulate geometric modeling as a mechanics simulation problem instead of as a mathematical geometry problem. For example, if we can simulate bodies of some appropriate modeling material, such as clay or foam, and we can present the hands of the user as bodies in the environment, all the user has to do is manipulate the material with his/her hands, presumably a familiar process. Our job then is to make sure the properties of the simulation fulfill the requirements of such a manipulation process. This application requires less adherence to physical principles, and we can for example introduce artificial operations, which have no real physical base, to take some load off the model. This type of application is normally referred to as “virtual claying”.

2 State of the Art

The mechanics of deformable bodies is in no way a new research area. There exists numerous models aimed at various applications.

At least one attempt has been made to augment a rigid body model with a special module for deformable body collision [Baraff, Witkin, 1992]. It applies a two-phase model, where the first phase prevents inter-penetration, and the second phase calculates contact forces. Deformations are constrained to what can be represented by a global deformation function, which avoids the problem of calculating impulse propagation. It is not clear however, how general this approach is. The authors also state that allowing complex deformation functions will lead to a heavy computational burden.

Particle models are often used where flexibility is needed with regard to the phenomena modeled. We use the term “particle model” to distuingish what the computer graphics community calls a “particle system” from the term used in physics. A particle model is a particle system with possible additional rules. Originally used to model smoke and fire phenomena, the models have developed to cover geometric modeling [Szeliski, Tonnesen, 1992]. Strictly speaking, most of the other models mentioned could be denoted particle models.

Another well-established model is the mass-spring model. Provot has described a model used for cloth simulation [Provot, 1995], and Chen et. al. a model aimed at general objects [Chen, et. al., 1998]. It describes bodies as sets of point-masses, and the materialistic properties as a graph of springs over these sets (essentially a particle model as well). While the model is very useful for describing deformation of a single body, it does not cover collision at all, since there exists no concept of volume. Computationally, it can give rise to stiff differential equations, for very stiff springs for example, which requires finer time discretization, and thus more

computation.

The primary tool for mechanics simulation in engineering analysis is called Finite Element Analysis (FEA) [Andersson, Sellgren, 1998]. However, there is inconsistency in the literature about the definition and scope of this term. The term is derived from the term Finite Element Method (FEM) of analysis. It is the term FEM which is inconsistently used.

In [Popov, 1999] (p. 104), the FEM is described as "More recently a powerful numerical procedure has been developed, where a body is subdivided into a *discrete number* of finite elements, such as squares or cubes, and the analysis is carried out with a computer". In [Heath, 1997], it is described as: "Finite element methods approximate the solution to a boundary value problem by a linear combination of basis functions ϕ_i, typically piecewise polynomials, which for historical reasons are called *elements*." The former definition is commonly used in engineering discussions, while the second is used in mathematical and numerical method texts.

The difference between the (informal) definitions, and what is causing the confusion, is that the first definition is a general discretization of a body, while the second is a method for solving boundary value differential equation problems, by discretizing the solution function in a particular way. Additionally, neither definition tells us anything about which physics model (what assumptions, etc.) is used.

Originally, and still principally, FEA refers to statics [Pedersen, 2000], and this is the definition we will adopt. A typical statics problem results in a boundary value problem, which can then be solved using the FEM. A basic dynamics problem on the other hand, results in an initial value problem, for which the FEM does not apply. When referring to an analysis method, it is preferable to refer to the physics aspect of the analysis instead of the numerical aspect. For instance, the FEM can be used to solve heat transfer problems, which have no relation to solid mechanics. To group such differing analyses under the term FEA causes ambiguities.

Terzopoulos, et. al. have developed a Lagrangian mechanics model aimed at animation [Terzopoulos, et. al., 1987]. They use continuous bodies, and a combination of boundary value (finite difference) and initial value methods. They treat elastically deformable bodies, and also collisions, which they handle by creating a force field around each body.

In [Terzopoulos, Fleischer, 1988], Terzopoulos and Fleischer extend this model with plasticity, and state an aim similar to ours: "We envision users, aided by stereoscopic and haptic input-output devices, carving 'computer plasticine' and applying simulated forces to it in order to create free-form shapes interactively". While we have not treated plasticity in our model, Their treatment of plasticity is likely to be directly applicable in future development.

Kang and Kak [Kang, Kak, 1996] have developed a FEA system to create a geo-

metric modeling system. The system presents the designer with an initial physical shape, represented by the FEM mesh. The user can then utilize a force-input interface, in their case a four-sensor plate, to manipulate nodes of the shape. The system could presumably be generalized to allow arbitrary input methods.

James and Pai use the Boundary Element Method (BEM) to create a virtual modeling environment [James, Pai, 1999]. This method is more suited to pure interactive deformation applications than the FEM due to only considering the boundary, and thus requiring less computation.

3 Mechanics Model

3.1 Basic Theory

We start building our theory at the atomic level. We know that any given body is made up of a large number of atoms, so if we can know the behavior of each atom, we will know the behavior of the body as a whole. An atom can be considered as a particle, a point mass. Between any given atom pair we have a central force, determined by the distance, and other properties of the atoms. This means we have a particle system, an entity which is quite simple to treat. If we have several bodies, we have several particle systems, which together simply can be treated as one particle system.

Now, we do not want to have an atom as the basic element in our model, to build any kind of useful bodies will require too many elements. However, we can make an approximation to overcome this. If we consider a solid body, it consists of many atoms close together, with neighboring atoms behaving in a similar way. If we were to treat every 2x2x2 matrix of 8 atoms in the body as a single element, we would end up with 8 times less elements to treat. These new elements would in turn form a particle system, and through induction, we could keep reducing the resolution until we have a manageable number of elements. For this to be possible, we need to show that we can approximate a 2x2x2 matrix of atoms, a particle system, as one single particle.

First of all, we need to formalize some of our statements. We have said that, in a solid body, “neighboring atoms behave in a similar way”. Formally, what we mean by this is that two neighboring atoms have the same properties, and that the external force, the force due to all other atoms in the system, on two neighboring atoms can be approximated as being equal. We then take this further to apply to a 2x2x2 matrix of atoms.

Particle system theory states that all forces that act on a particle in a particle system

can be divided into two sets: internal forces, which stem from other particles in the system, and external forces, which stem from outside the system. This distinction is made because internal forces balance out, and do not affect the center of mass of the particle system. Thus, the center of mass motion can be determined strictly by taking external forces into consideration.

If all particles have the same external force applied to them, and all particles have the same properties, the result on the center of mass will be the same as if we treat the system as one particle, with the sum of the external forces applied to it, and with the sum of all the properties added to it. It is also clear that with this arrangement, all torques with regard to the system's center of mass cancel out, and the angular momentum of the system is conserved.

What we have done with this approximation is to remove resolution from the description of a body. Since an element of the body must by our definition be homogenous, we cannot have different forces acting on different parts of the element, as would be the case if the element was decomposed into its original parts. We will have to take this into consideration when we consider what kind of forces are acting between two elements. For example, interatomic forces have components which only are significant for a very small distance. If our elements are significantly larger than this distance, our approximation errors will be very large. We can however compensate for this by trying to find some sort of average force over the entire element. However, we have not examined this topic very closely yet, and our force models are still very simplified.

3.2 Model Description

In the previous section, we describer how we can reduce the complexity of the atomic configuration of a body into larger and fewer "elements". We now need to describe how such elements interact, and more formally describe the components of the model.

We started off by examining how atoms are configured in a body, and we need to apply the same reasoning to determine how our elements interact. Since we have removed much of the resolution from the configuration of elements, we introduce an entity for interaction which allows us to retain some of the complexity. We say there is a "connection" between two elements when they are interacting, and that such a connection has some state associated with it. We are also free to determine when an element pair will start interacting, and when it will stop. This can allow us to overcome some of the lack of geometric shape of an element.

With such an entity, we can define a force between a pair of elements which is not only dependent on some instantaneous distance or parameter of the elements, but also on some parameter of the connection, which we can deduce through other

means. For example, although the contact force between two bodies and the strain force inside a body both physically stem from the same inter-atomic forces, the structure of the atoms inside a body might be different from that seen in the interaction between two bodies. We could describe that using our connections, thus creating a different force function between elements known to be fused to one another, and elements of different bodies.

Based on what we know of the micromechanics of inter-atomic behavior, and of some of the more macroscopic mechanics of bodies, we have formulated several forces we can use in our model:

Gravitation

First of all, we have a gravitational force. If necessary, we can describe a detailed gravitational force model where each element determines the force on every other element. However, normally it is sufficient to have a uniform gravitational force, where only one mass is the source of a gravitational field.

$$F_g = G\frac{m_1 m_2}{r^2} \tag{1}$$

G gravitational constant
$m_1,\ m_2$ masses
r distance

Elasticity and Fracture

To determine inter-element forces, we start off by examining a graph of the interatomic force (See figure 1 [Kleppner, Kolenkow, 1978]). We can see that there exists a distance where the force is zero, and that the force becomes repulsive when decreasing the distance, and attractive when the distance is increased. If the distance is increased beyond a limit, the force decreases to insignificance. We can model this by using a simple Hooke formulation, and two threshold distances. One distance determines when two elements are close enough so the force is significant enough to be taken into consideration, and another determines when two elements have receded far enough from each other to no longer significantly interact.

$$F_c = -k(d - l) \tag{2}$$

d actual distance
k Hooke constant
l nominal distance

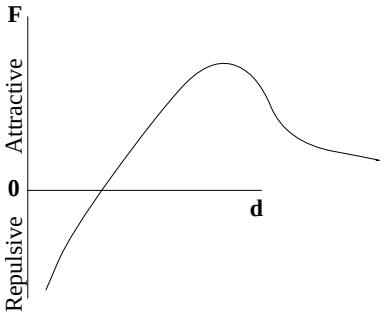

Fig. 1. Sketch of the inter-atomic force function

We now have the core model. See figure 2 for a schema of two elements, a connection, and important quantities. We can now construct arbitrarily shaped bodies, in arbitrary initial states, and simulate the behavior. We will see however, that such systems do not behave very well. In reality, most actions and interactions inside and between bodies involve nonconservative processes which damp the motion of such systems. As no such processes exist within our model, the systems will simply oscillate eternally, or more probably, if solved numerically, oscillate divergently and "explode".

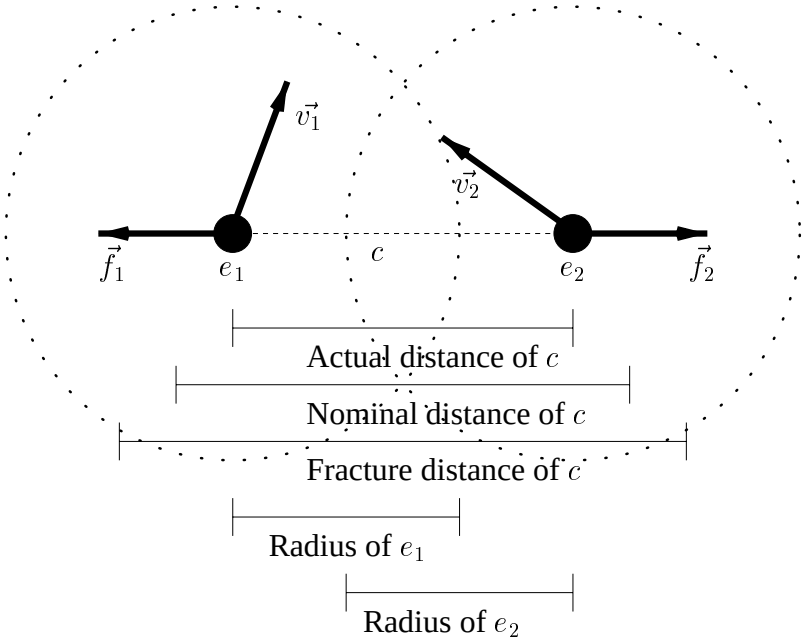

Fig. 2. Schema of two elements, a connection, and important quantities.

Therefore, we need to identify, and find a way to incorporate such processes into our model. Unfortunately, such processes are not as simple as what we have seen so far, and need more heuristic and approximative methods.

There are three easily identifiable nonconservative processes we can observe: internal damping (compress and release a foam ball), sliding friction (run and fall down) and ambient viscous friction (throw a foam ball into the air at high speed). However, there does not exist any simple general models for these processes, the

models that exist are only based on very special cases. Regardless, as long as they convey a reasonable approximation of the process, they are useful.

Internal Damping

We damp the linear inter-element force with viscous friction, oriented along the elongation velocity vector:

$$F_d = -b \, \|\vec{e}\| \tag{3}$$

b damping constant
$\vec{e}$ elongation velocity

Sliding Friction

We use the standard sliding friction model, oriented along the velocity vector component normal to the normal force vector:

$$F_f = -\mu \|\vec{N}\| \tag{4}$$

$\vec{N}$ normal force
μ friction constant

While the empirically found friction constants are only valid for interaction between two specific materials, we simplify this a bit, and define a friction constant for every element. When two elements interact, we average the constants for the friction force between the elements.

This force is separated from the other forces in that it is not central. We have to be careful when using it, because in conjunction with our approximation, it can lead to unexpected behavior. For example, in a body which is compressed and rotated, "sliding" occurs between internal elements, producing a sliding friction force. While this behavior does not violate the model, it may not be what is expected.

Ambient Viscous Friction

We use the model for fluid resistance at high speed, oriented along the velocity vector:

$$F_v = -\pi r_e^2 \rho \|\vec{v_e}\|^2 \tag{5}$$

ρ medium density

Formalized Definitions

Element:

An element e is a set of parameters $\{ \vec{p},\ \vec{v},\ b,\ m,\ r,\ k,\ t,\ \mu,\ C \}$.

$\vec{p}$ position
$\vec{v}$ velocity
b damping constant
m mass
r radius
t fracture distance
μ friction constant
C set of connections connected to the element

Connection:

A connection c is a set of parameters $\{ e_1,\ e_2\ b,\ k,\ l,\ t,\ \mu,\ \}$.

$e_1,\ e_2$ elements comprising the connection
b damping constant
k Hooke spring constant
l nominal distance
t fracture distance
μ friction constant

Connections are dynamically created and destroyed when elements start interacting and stop interacting respectively. The parameters of a newly created connection are calculated from the parameters of the elements it connects. This is why the element primitive share some parameters with the connection primitive. Exactly how the new parameters should be calculated remains to be determined. Presumably, the radii of the colliding objects should also be taken into consideration. For now however, we simply average the respective parameters for the new connection.

We are now satisfied with the components of the model. We can reasonably correctly simulate most phenomena observed in systems of deformable bodies. We will now proceed to show how we can numerically solve such systems defined within this model.

4 Implementation

4.1 Numerical Solution

The nature of the inter-element forces in the physical model may provide for difficulties in mathematical treatment. Since we externally control the force functions, we control the connection entities through a state machine, as the system develops over time, formally we should include this state machine in our functions. How-

ever, piecewise in time, the state (the connections) remains the same, and we do not have to take this into consideration. Thus, we can describe the system as a series of differential equation systems, where the initial state of each system is determined by the state of the previous system, and by a state machine.

The mathematical formulation for each single system is quite simple. We have a standard particle system which we want to develop in time.

We can define the force on a single element e in the system:

$$\vec{F_e} = \vec{F_C} + \vec{F_G} + \vec{F_L} \tag{6}$$

$\vec{F_C}$ the sum of all the connection forces
$\vec{F_G}$ the sum of all global forces
$\vec{F_L}$ the sum of all "local" forces, i.e. forces depending only on the state of the element itself

We then use Newton's second law, $F = mp''$, to produce a system of second-order ordinary differential equations:

$$\vec{p_e}'' = \frac{\vec{F_e}}{m_e} \tag{7}$$

To simplify solution, we want to transform our system into a new system of only first-order ordinary differential equations. We define two new vector functions:

$$g_1 = \vec{p_e} \tag{8}$$

$$g_2 = \vec{p_e}' \tag{9}$$

We now have a new system:

$$g_1' = g_2 \tag{10}$$

$$g_2' = \frac{\vec{F_e}}{m_e} \tag{11}$$

Thus, solving (11) produces g_2, which we can use in (10) to produce g_1, which is equal to $\vec{p_e}$, the solution to (7).

We can now solve this system using our preferred numerical method. We have to keep in mind that mathematical treatment may encounter difficulties however, due to the usage of this state machine. As a start we choose Euler's method. It has proven to be practically usable, so even if we never manage to apply any other methods, we can still create a practical implementation.

The required stepsize is dependent on a number of factors. We are not so much interested in correctness as in stability. We can reason that the Euler method (as well as most other numerical methods) works by sampling the state of the system, and then extrapolating current state to produce the next state. Since the extrapolation necessarily will bound state changes in our system, such as a collision, we have to make sure we do not extrapolate too far. This means that the step size is dependent on the velocity, position and radius of the elements in the system. Since our method is not adaptive, we must choose a stepsize in advance, which will correctly handle the most extreme event in the simulated sequence. Step size will also be dependent on the stiffness of our connections. This is however related to the previous attributes (velocity, etc.), as a connection only can provide an acceleration.

As we specify the initial state of the system when we describe what we want to simulate, we have all the information we need to start the Euler iteration. All we now need to do is to define $\vec{F_e}$ formally. We define E as the set of all elements in the system. The subscript e refers to the element we are calculating the force on. We also define an informal order in the set so we can iterate through it. We define each component separately:

Local Forces

(ambient viscous friction is the only force)

$$\vec{F_L} = -\pi {r_e}^2 \rho ||\vec{v_e}||^2 \frac{\vec{v_e}}{||\vec{v_e}||} \tag{12}$$

Global Forces

(gravitation is the only force)

$$\vec{F_G} = \sum_{i=0}^{|E|} G \frac{m_e m_i}{||\vec{p_e} - \vec{p_i}||^2} \frac{\vec{p_e} - \vec{p_i}}{||\vec{p_e} - \vec{p_i}||} \tag{13}$$

(Normally we only define one body as a gravitational source, to reduce computation, or a uniform gravitational field)

Connection Forces

As the friction force depends on the other components of the inter-element force (which define the normal force of the friction equation), we need to further subdivide the inter-element force. For clarity, we simply use the logical components we have already defined:

$$\vec{F_C} = \vec{F_b} + \vec{F_d} + \vec{F_f} \tag{14}$$

$\vec{F_b}$ original inter-element force definition
$\vec{F_d}$ damping force
$\vec{F_f}$ friction force

To simplify notation, we define the subscript e as we have done before, and a new subscript p as the opposite element in the connection. We iterate over the connection set C of the element:

$$\vec{F_b} = \sum_{i=0}^{|C_e|} -k_c(||\vec{p_e} - \vec{p_p}|| - l_c)\frac{\vec{p_e} - \vec{p_p}}{||\vec{p_e} - \vec{p_p}||} \tag{15}$$

We define the relative velocity of the two elements in the connections as two components, one parallel to the connection, and one orthogonal:

$$\vec{v_{||}} = \frac{(\vec{v_e} - \vec{v_p}) \cdot (\vec{p_e} - \vec{p_p})}{||\vec{p_e} - \vec{p_p}||^2}(\vec{p_e} - \vec{p_p}) \tag{16}$$

$$\vec{v_\perp} = (\vec{v_e} - \vec{v_p}) - \vec{v_{||}} \tag{17}$$

Then:

$$\vec{F_d} = \sum_{i=0}^{|C_e|} -b_c(\vec{v_{||}}) \tag{18}$$

The sum of these two forces could be called “contact force” in certain contexts. They form the normal force in the friction definition,

$$\vec{F_N} = \vec{F_b} + \vec{F_d} \tag{19}$$

We can now define the friction force:

$$\vec{F_f} = \sum_{i=0}^{|C_e|} -|u_c \vec{F_N}| \frac{\vec{v_\perp}}{\|\vec{v_\perp}\|} \tag{20}$$

4.2 Algorithms and Performance

Before we discuss performance and efficient algorithms, we make some assumptions about the state of the system to remove the need to treate degenerate cases.

The state of the system consists of a set E of elements, and a set C of connections. The set of connections form a graph over the set of elements. For $|E| = n$, we have that the minimum of $|C|$ is 0 and the maximum is n^2. However, for the applications we have in mind, we make the assumption that there exists a structuring on the elements so that they are well-separated. This means that any given element has a number of connections which can be bounded by a constant independent of n. For instance, according to our original argumentation about elements, we formed elements from a 3D matrix of smaller elements. If we create connections between vertical, horizontal and diagonal neighbors in the matrix, we end up with $3 \cdot 3 \cdot 3 - 1 = 26$ neighbors, which is also the number of connections per element. Since each connection consists of two elements, we will have $13n$ connections for this example. During simulation, this may not be true locally, but for non-degenerate situations, we should be able to find a constant which can bound the number of connections.

Force Calculations

The force calculations consist of simply performing the arithmetic as dictated by our force definitions. We have two kinds of force calculations, one which calculates a component of the force of a connection, and one which calculates a force component of a global force. Thus, for each connection force component, we have to make one calculation per connection, and for each global force component, we have to make one calculation per element. According to our previous assumption of well-separatedness, both of these are $\Theta(n)$.

Numerical Integration

Numerical integration consists of calculating a new state of the system for every time step, as has been described. We have one equation per element, so this algorithm also is $\Theta(n)$.

State Machine Operations

The operations by the state machine so far only include determining when connections should be created and destroyed, and what properties they should have. This is a more complex computation. To determine when a connection should be destroyed, we simply have to perform a test for each connection, and if its elements are a distance t apart, we destroy it. Thus, this operation also is $\Theta(n)$. However, the opposite operation does not. Given a distance r for every element, we create a connection when two elements are within $2r$ from one another (and a connection does not already exist). The brute force algorithm compares every element with every other element, so this algorithm is $\Theta(n^2)$. We will see we can do better than that.

This problem can be formulated as the well-known collision detection problem. Given a set of shapes, find all pairs which intersect. Although we can never find an algorithm better than $O(n^2)$ in the general case, since there might exist $O(n^2)$ collision pairs, and we somehow need to find them, we can isolate such cases. There are several algorithms which are significantly more efficient than the brute force algorithm. One is based on dimension reduction [Cohen, et. al. 1995], and is $O(n \lg n + m)$, where m is the number of shape pairs which are "very close". We can also apply hierarchical space partitioning [Jansson, Horváth, Vergeest, 2000], which while difficult to prove, empirically performs as $O(n \lg n)$.

Global Performance

If we add our complexities together, we get $O(n \lg n)$ (the most expensive complexity). We have performed empirical experiments which indicate this behavior [Jansson, Horváth, Vergeest, 2000]. For absolute performance, we refer to the same paper. We show that up to ca. 500 elements, with semi rigid material properties, we can achieve real time performance on standard PC hardware.

4.3 Creating Bodies

Before we can do anything practical with the model, we need to specify the state of the model, i.e. the elements and connections and their properties. Our aim is to be able to take a standard solid CAD model as input to our system. If we look back at how we defined an element, we can see that it is quite straightforward to translate a

given geometric description of a solid body into the physical representation of the model.

The geometric description of a solid body defines the volume of the body. The volume of a body is simply the union of all the atoms in the body. Since an element in our model simply is a spherical approximation of a large number of neighboring atoms, we can easily create a translation.

First of all, we need to decompose the geometric description into polyhedra, each which must be approximable by a sphere. These form our elements. We then determine the topology of the spheres from the topology of the polyhedra. From this topology, we can determine which spheres are connected. If we assume the body initially is in its rest shape, we specify the nominal distances of the connections so that there is no strain energy, concretely, we specify the nominal distances to be the actual distances between the elements. Since a geometric description has no physical attributes, we cannot determine any other parameters of the elements or connections from this description alone, but need extra information.

In our implementation, we can translate solid polygonal representations into the physical representation of the model (any solid representation should be translatable with this method). First of all, we convert the polygonal representation into a voxel representation by sampling the polygonal representation with an inside/outside function. We then generate an element for each voxel, with a diameter no more than twice the voxel width (so two neighboring elements can intersect, but no further). Connections are generated depending on the topology of the voxel matrix. We say that the neighbor of a voxel is any voxel which shares a vertex, and a connection is created for each neighbor. Figure 3 illustrates how four voxels have created four elements (with shrunk diameters to prevent occlusion), with connections.

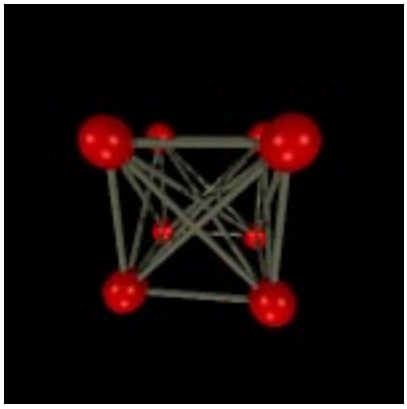

Fig. 3. Illustration of how four elements are interconnected (size of elements have been shrunk to prevent occlusion).

We can demonstrate a practical example. Say we have a boundary description of a part as described by figure 4. We can then generate a physical description using the

previously described method to generate the description as shown in figure 4.

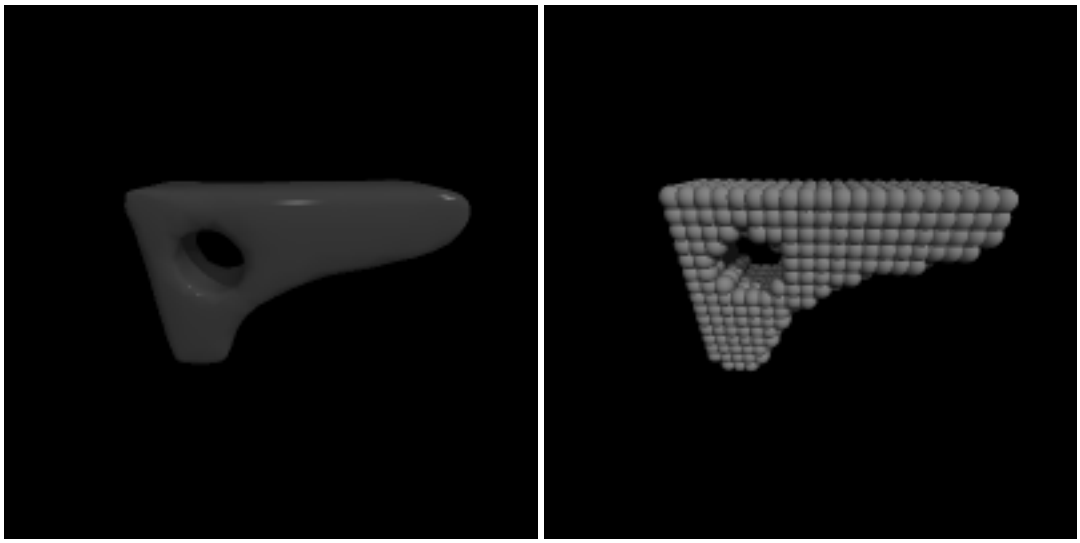

Fig. 4. Boundary and physical representation of a part (a support).

4.4 Interface

When practically using the model, it is normally not enough for the model to be isolated, we somehow need ways to interface with it, either interactively in real-time, or with other simulation systems. If the model publishes a standard interface, we can create a modular system, where models and devices easily can be replaced as needed.

4.4.1 Manipulation

We define two different ways of manipulating the state of the model: *physical* and *artificial*.

A physical manipulation is indirect, the manipulator needs to create entities in the model, and then use those to perform the manipulation. An example could be that the manipulator creates a tool as a configuration of elements and connections, and then applies that tool to some body in the environment to perform a deformation.

An artificial manipulation is direct, the manipulator changes the values in the state directly, thus bypassing the physical laws of the model. For instance, to control the previously discussed tool, a predefined path could provide values which are given the positions of the elements as time progresses.

4.4.2 Human Interaction

With human interaction, we mean a real-time interaction which feels natural to a user, as if the user is interacting with something in the real environment. This can be supported to varying degrees, we will try to describe the relevant components.

At the most basic level, the user is a body which we want to represent in the model. The information which flows from the user to the model is position information of the body. The information which the model can provide back is position information of the bodies in the environment, but also force information where the user's body is interacting with bodies in the model.

This interaction has to be handled by actual physical devices. Position and orientation information can be sampled in many ways, through electromagnetic sensors or mechanical arms for example [Berkley, et. al., 1999] [Bullinger, et. al., 1999]. In our testing, we have used a simple mechanical arm (see figure 5) which can sample position and orientation of a rod. Visual feedback can be produced by a computer monitor, also in stereoscopic form, and with a computer graphics visualization of the state of the model. There exists devices for force feedback, or haptic feedback, but such devices are still not as established as the previous types. We have not had the opportunity to perform testing with such devices, this will be a future field to explore.

Interfacing the model with a haptic device should be fairly straightforward. We represent the haptic device as a body, whose position and orientation is provided by the device. When the body interacts with other bodies in the environment, there will be forces acting on the "device body". These forces can then be displayed by the haptic device, and will thus be translated into the real world, where they will accelerate the "device body" in reality, and thus change its position. This creates a loop of feedback. As mentioned, we have unfortunately been unable to test this.

5 Preliminary Verification

While we have a reasonably sound theoretical foundation, it is important to verify the model experimentally. We have performed some limited experiments which at least can give us an indication.

The experiment setup is a beam (see figure 7, the right image), fixed at one end, with a static perpendicular load applied at the free end. The beam is $2m$ x $0.7m$ x $0.5m$, the density is $0.58E3\frac{kg}{m^3}$ and the elasticity modulus is $12.1E5\frac{N}{m^2}$. There is no gravity. The simulation is run until there is equilibrum (within a threshold), and the displacement of a point at the free end is measured. According to beam theory in mechanical engineering [Popov, 1999], the displacement grows linearly

Fig. 5. We have used a MicroScribe 3D device to track position and orientation in real time.

with the applied load, for small displacements. Thus, we apply a range of loads and analyze whether the relation is in fact linear. For now, we do not examine whether the magnitude of the displacement for a given load and material correponds with beam theory, we are only interested in the relation.

In the graph at the bottom of figure 6, we have plotted the results of the experiment. We can directly see the relation is linear.

We have also examined the impact of resolution on this particular experiment. We take the same beam, but double the resolution (see figure 7, the left image), and examine the result. Before we can do that, we must use the same material in both beams, or the result will be meaningless. Ganovelli et. al. [Ganovelli, et. al., 2000] present a relation between the Hooke constant, Young's modulus, the spring length, and the volume of the tetrahedron which the spring mesh forms:

$$k_i = \frac{EV(\tau)}{l^2} \tag{21}$$

τ Tetrahedron considered
E Young's modulus
V Volume
k_i Hooke constant contributed by tetrahedron
l spring length

However, this relation has not provided consistent results for us. This will have to be examined more deeply in the future. Since at this point, we're only interested in

the relation between two beams, we use a similar relation, which is still consistent with unit analysis:

$$k = El \tag{22}$$

E Young's modulus
k Hooke constant
l spring length

We used this relation for the previous experiment, and now we perform the same experiment, but with twice the resolution. In the graph at the top of figure 6, we have plotted the results of the experiment. We can again directly see the relation is linear, and also that the graph is very nearly identical to the bottom graph. This means the behavior of the beam in this experiment is resolution-independent. As before, the absolute numbers are not important for this particular aspect to be proven. Due to the discretization process, the two resolutions do have differences, in mass and geometrical extent for example, but these differences are minor.

6 Applications

We now have a quite general model. The aim is now to determine how to apply this model, and in which applications it can be advantageous to use this model, compared to existing methods.

We can divide possible applications into two fields: *interactive* applications and *offline* applications. The main difference is that there exists a much tighter time constraint on interactive applications.

6.1 Geometric Modeling

Geometric modeling is a typical interactive application. During shape design, geometric modeling is used both as a creative tool (sketching, claying), as well as for the final description of the shape. Normally, these two processes are not integrated. We will describe a geometric modeling application which integrates the two processes.

Geometric modeling can be done through a physical interface. We define a number of bodies as manipulators, which the user controls to manipulate other bodies in the environment. This provides a completely natural interface, and requires no knowledge of the model. However, due to limitations of the model, such a system

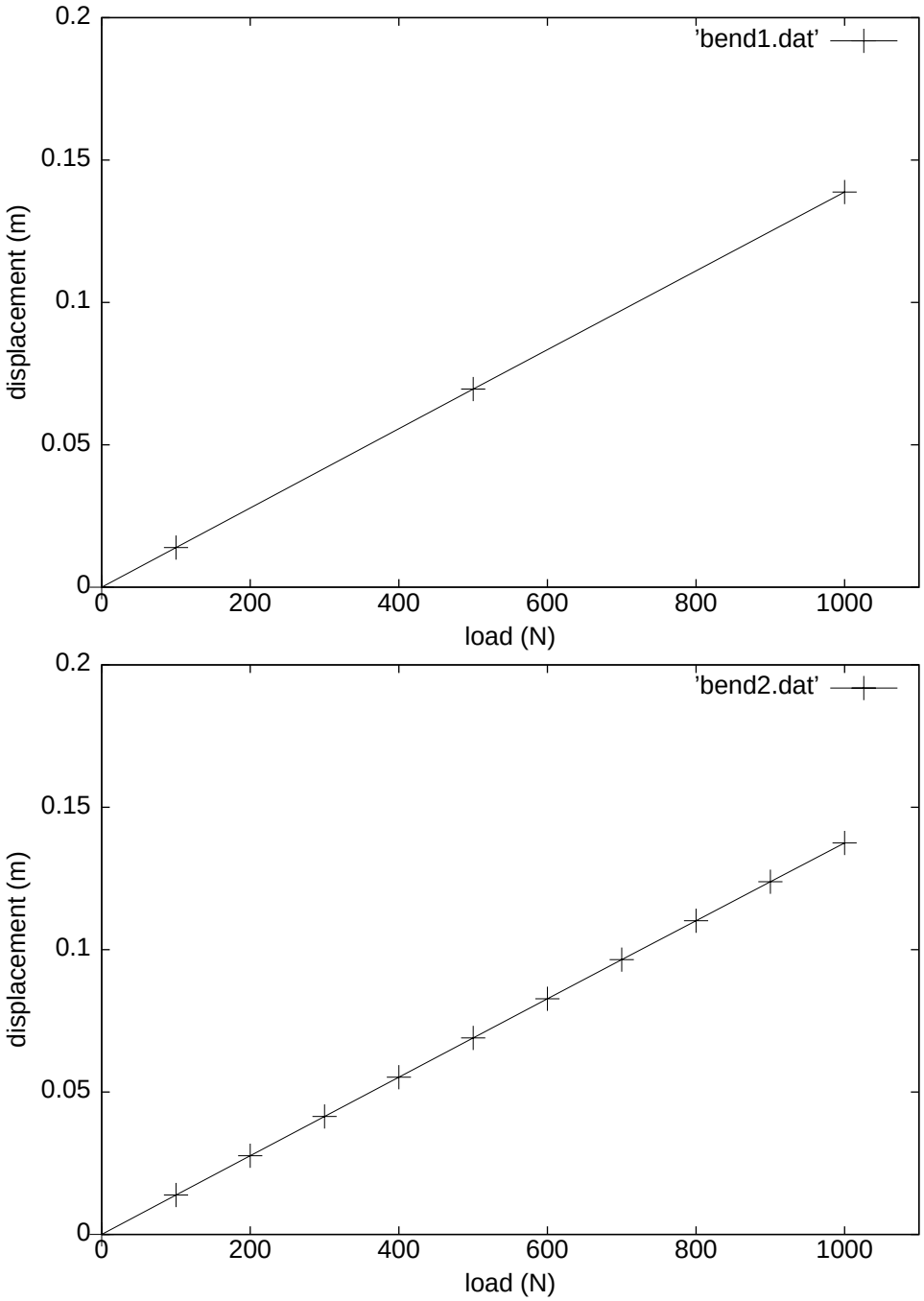

Fig. 6. Graphs describing the relation between the perpendicular loading of a beam and its displacement. The beam in the top graph is of twice the resolution than the bottom.

is not yet fully practical. For example, since the model only supports elasticity and fracture, and no plasticity, it is not possible to perform a permanent non-fracturing deformation on a body. Therefore, we need to introduce a number of artificial operations, which can provide a replacement for full plasticity, as well as other properties.

We also have to make the distinction between deformation and creation/annihilation.

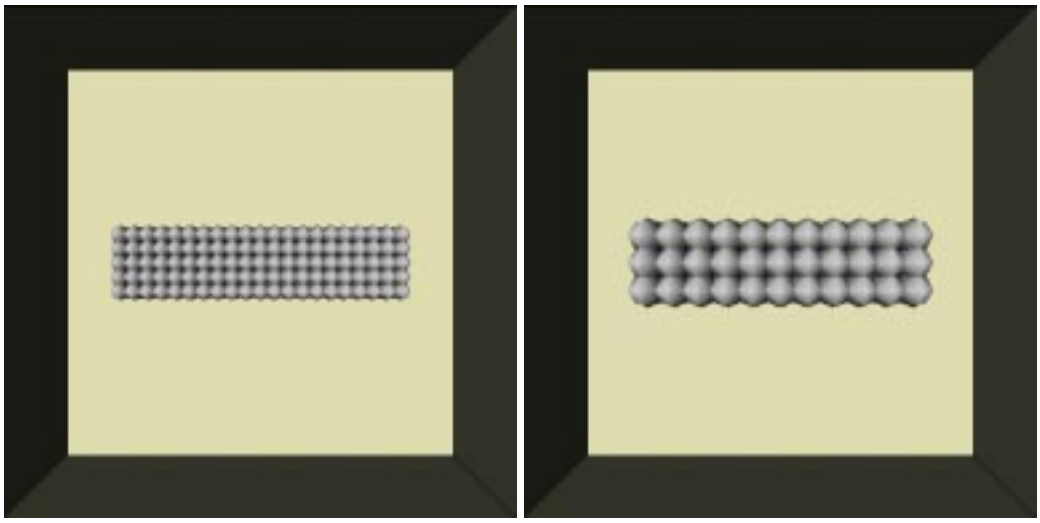

Fig. 7. A beam in two different resolutions.

The model supports deformation but not creation/annihilation of bodies or of elements of bodies. However, this could be resolved in a similar manner. We could simply couple the model with a system able to create/annihilate geometry, and then perform a translation procedure between the two systems. In our tests, we have used standard and custom geometric modeling packages to create basic shapes, which are then translated into a physical representation, and imported into the environment. We have not attempted annihilation as of yet.

6.1.1 Modeling Operations

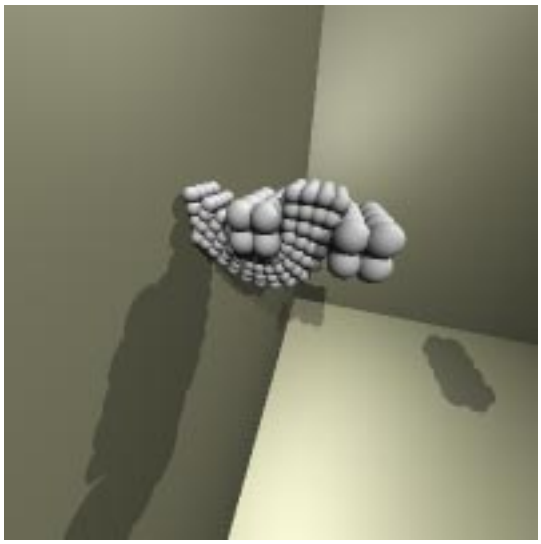

Fig. 8. The manipulators are applied for a bending operation.

(1) Physical Deformation

The main operation is simply interacting with the physical model through contact forces, generated by the interaction between the manipulators and the

bodies in the environment (See figure 8). With this operation, we can also perform fracturing, though such an operation might be implemented more efficiently as an artificial operation. Depending on how we interpret the state of the model, we could also perform topology changes with this operation (aside from fracturing, which directly can perform topology changes).

(2) Renormalization (artificial plasticity)

Since we have no plasticity, deformations are essentially useless for prolonged modeling since they are not permanent. We can alleviate this by introducing an artificial operation called "renormalization". When this operation is performed on a certain body, all connections in the body assume the current distance as the nominal distance. Macroscopically, this means the body takes on the deformed shape as the rest shape.

(3) Direct Attribute Modification

A generalization of renormalization is "direct attribute modification". Given that we can select a part of a body, or a body, and determine which elements comprise the selection, we can artificially modify the attributes of the elements or of the connections between the elements. We could for example make part of a body stiffer, so that deformation of the entire body has less impact on that specific part. Another possibility could be increasing the fracture distance of elements of parts of a body, thus making it more "sticky". This could be used to glue parts of bodies together, or, applied to a manipulator, could increase flexibility of manipulation. However, the only such operation which has been implemented and tested is this "renormalization". We have also tested the "glue on manipulator" concept, but only as part of defining the actual manipulator.

6.1.2 Deformation Mapping

While such a geometric description is enough, when a b-rep (boundary representation) description already exists, it can be useful to use that representation directly, and then map the deformation of the physical representation to the b-rep. We have previously described how we can generate a physical representation from a b-rep. If we store the b-rep with the physical representation, we can use the b-rep for visualization and post processing, while using the pbysical representation for simulation.

There exists methods which can map deformation of a "cage" to a b-rep shape inside the cage [Sederberg, Parry, 1986]. We can apply a similar method, but locally for each vertex of the b-rep, and let the cage be defined by neighboring elements. Normally, interpolation is used to determine the deformation inside the cage, however, since our cages are very local to the vertices, we do not use interpolation.

We denote the space where the elements are expressed as "simulation space". For each vertex in the b-rep, we find four close elements which we can generate an orthogonal base from, using the Gram-Schmidt method for example. One element

forms origo, one element forms the primary axis, while the other two are used to determine the orientations of the two secondary axes. We then transform the vertex into this new base, and store this representation. If we now transform the vertex back into simulation space, we will get back the original vertex. However, if the simulation has led to a deformation of the original positions of the elements, the vertex will also be deformed according to the deformation of the space determined by the elements.

We can view this using a Voronoi diagram formulation. If we generate the Voronoi diagram of the elements, we will end up with cells, where each vertex of the b-rep exists in a cell. If the elements are deformed, the cells are also deformed. This way, we can determine the discrete space deformation from the deformation of the elements. However, this model is only valid for one specific Voronoi diagram, so deformations which give rise to new diagrams may produce erroneous results. Depending on the tolerance of the application, such errors may or may not be acceptable. In our testing, such errors have been acceptable when only visualization is required.

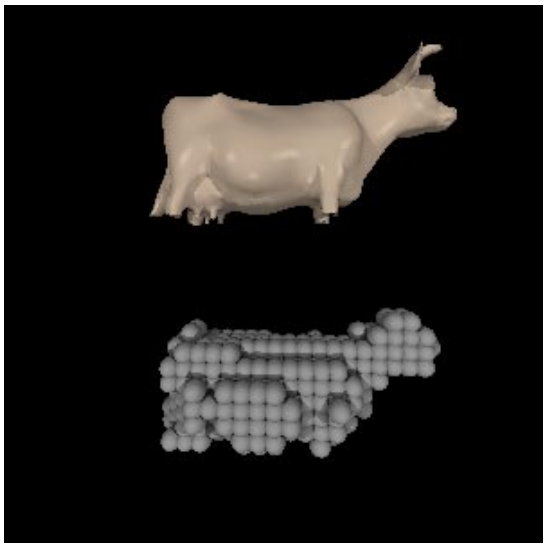

Fig. 9. A b-rep of a cow shape is translated to a physical representation.

6.1.3 Geometric Modeling Example

See figure 11, combined with the previous figure 8 for illustrations how we can perform operations using a natural interface to produce arbitrary deformations.

6.2 Analysis

As mentioned in the introduction, it can be useful to perform simple analysis in the conceptual stage to determine fruitful design directions. It is obvious that if the

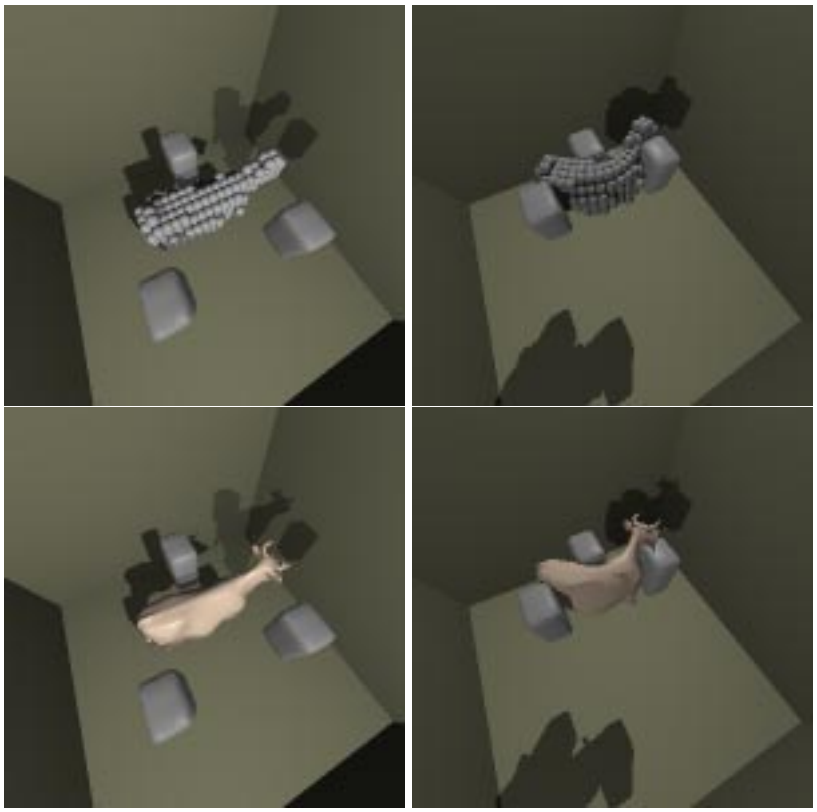

Fig. 10. The deformation of the physical representation can be mapped onto the b-rep.

later stage analysis could be performed at the conceptual stage, it would be done, however, this is not practical. What we instead have is a cheaper variant, which produces less accurate results. However, it is better to have some results which can be indicative, and later can be verified more thoroughly, than to have none at all, and perhaps have to go through a redesign after later analysis.

6.2.1 Analysis Example

Our example considers the analysis of a design of a support part. We fix two supports by the back surfaces, and then let them support a thick beam. We then drop a heavy cylinder on the beam, and examine the behavior of one of the supports during impact. The supports and beam are semi-rigid, perhaps comparable to a wood material, or a polymer.

Figure 12shows the system during the simulation. Figure 13 shows a close-up of only one of the supports during simulation, with everything else in the system removed from the visualization. Since we cannot yet map materials from the real world into the model, such analysis is not yet practically usable for most cases. However, if the material properties can be made to match reasonably, through empirical testing for example, it can be used to compare behaviors of different designs.

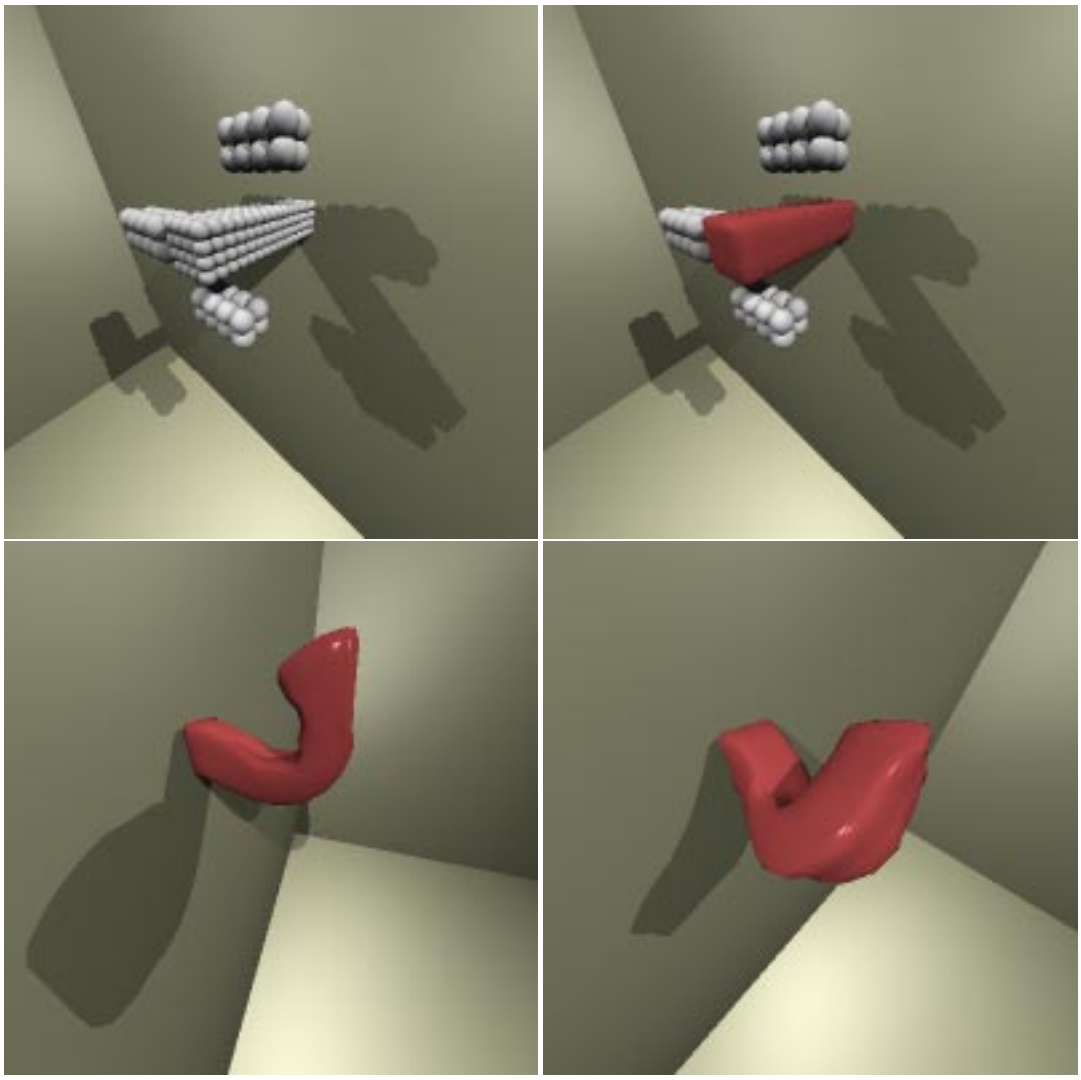

Fig. 11. We start with a bar, which we want to perform bending operations on. By applying tools, we can perform bending operations on the bar. We are however limited by the simplicity of our manipulators, and the interface. We use our artificial plasticity to first bend the bar upwards, then “freeze” that shape as the nominal shape. We then bend the deformed bar sideways.

The support, beam and cylinder were modeled using a traditional solid modeling system (Rhinoceros). The translation of the geometry required an insignificant time, however, some manual input had to be made (element sizes and material parameters). The total computation time for the simulation in the example was less than 600s. The hardware used was a dual processor (Intel Celeron, 450MHz) PC-AT system.

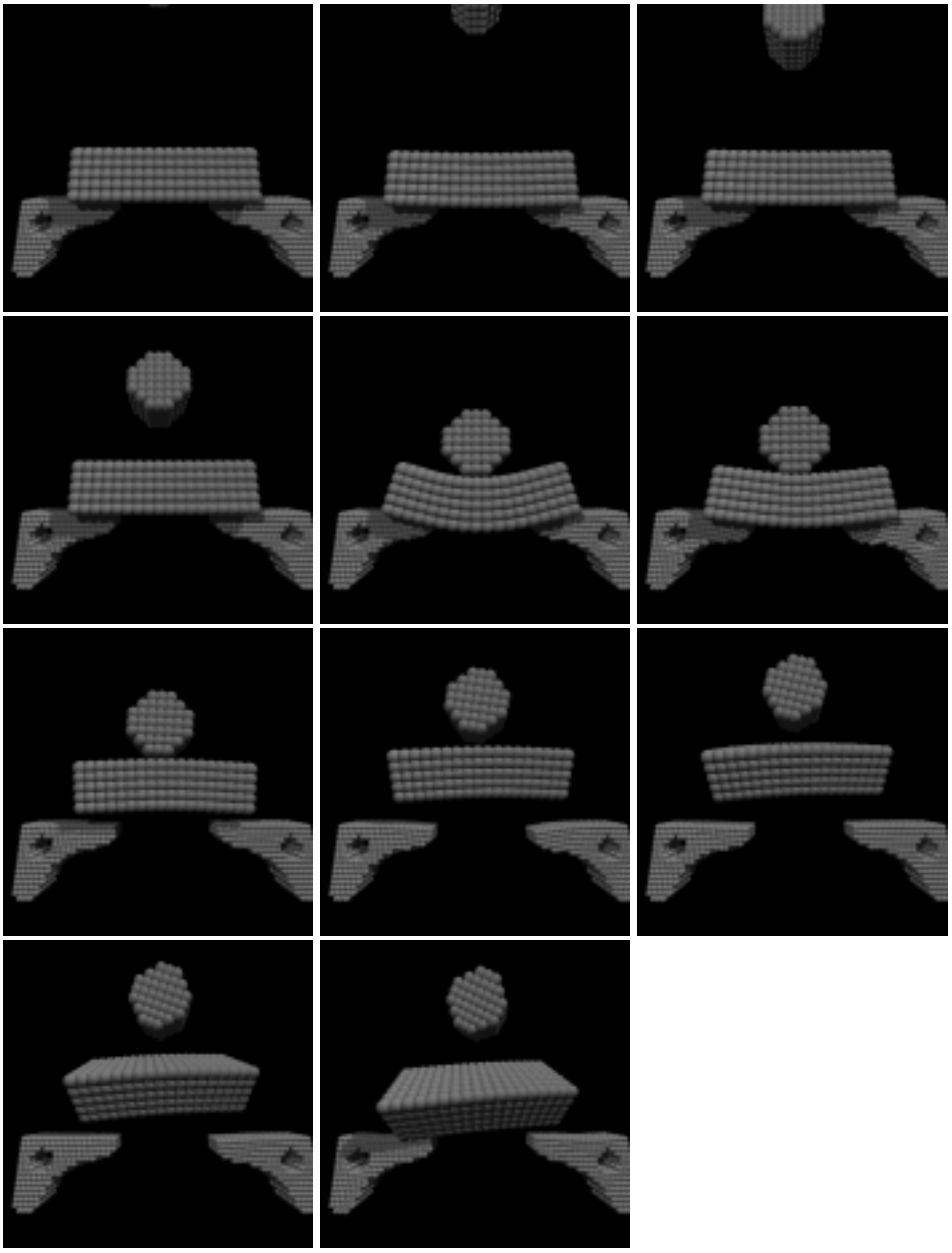

Fig. 12. The beam and supports are initially at rest, while the cylinder is falling rapidly. The cylinder impacts on the beam, and the beam and supports flex quite significantly. The supports and beam flex back, and launch the cylinder upwards again, the beam is also launched. The total computation time for this simulation was less than 600s.

6.3 Integration With Existing Methods

After conceptual shape modeling and analysis has been performed, and some possible designs have been produced, we need to transfer this information to detail design systems, and perform more accurate analysis. However, this transfer is theoretically trivial, since we at least implicitly have a discrete boundary representa-

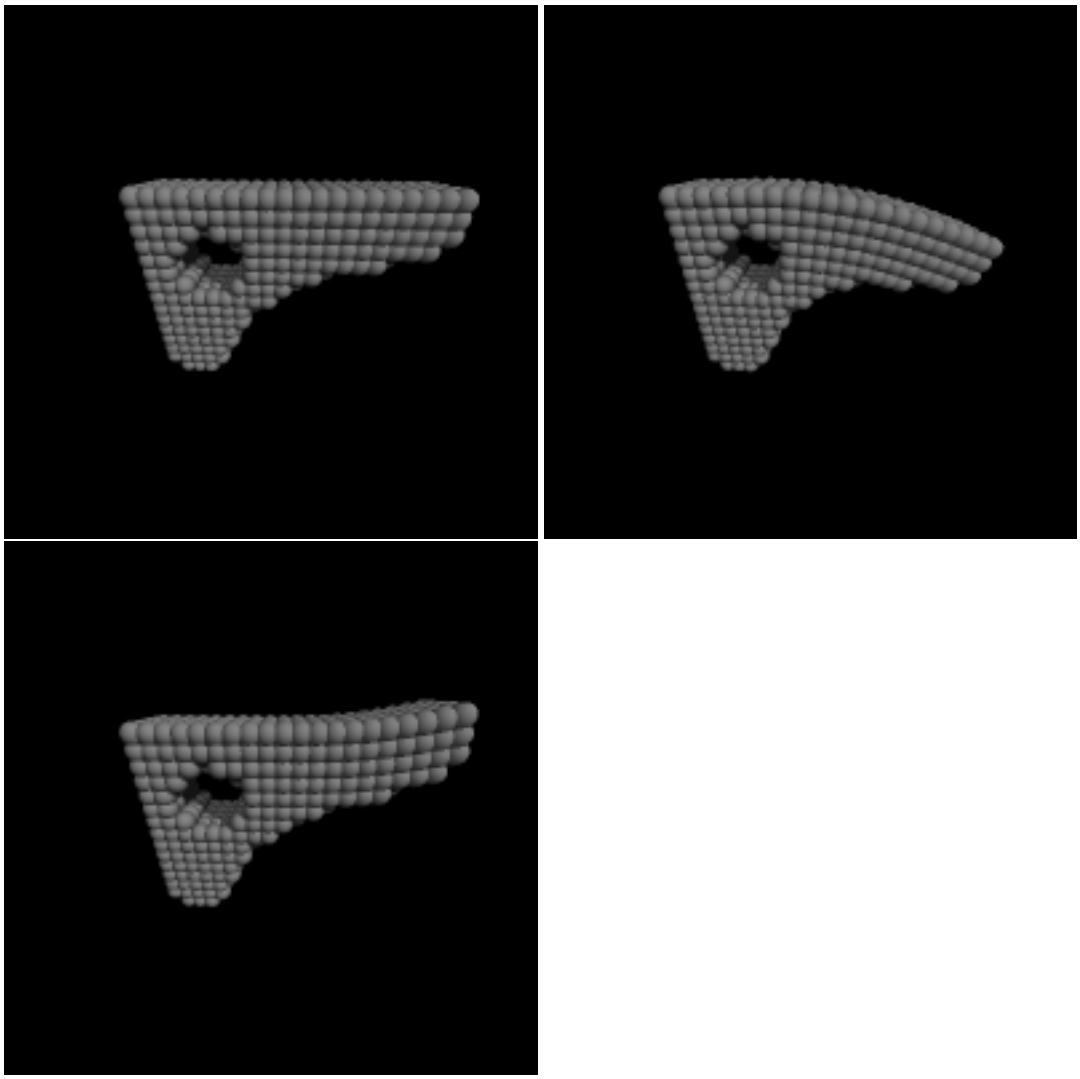

Fig. 13. A close-up of one of the supports. In the first image the support is at rest. In the second image the impact is at its extreme. In the third image the support has flexed back. Aside from a large deflection, we see no anomalies in the support.

tion of the geometry of the bodies, and a discrete representation of the physical quantities of the bodies.

More difficult, and interesting, is direct integration with existing methods. For example, we might have parts of the design which are at the conceptual phase, and parts which are already detailed. An important future aim could be to be able to perform simulation with both a traditional Finite Element Analysis model and a model such as this directly interfaced, so any two bodies from different representations could interact.

The presented model is meant to be used in conceptual design support tools. This means that it has no direct influence on the structure of the design process, it only enhances certain stages. Any design process normally consists of first creating a concept accoring to specifications, and then analyzing whether it actually meets those specifications.

This model can be used at the concept shape creation stage, as a kind of "virtual claying", or more natural geometry manipulation than typical direct geometrical entity manipulation. At the analysis stage, it can be used to give an indication of feasibility, or a rough estimate of the behavior. An analysis example could be a concept for a new type of washing machine. This type of simulation could then indicate how much angular velocity of the drum is needed to create the desired friction or movement of the cloth, and also what vibrations and deflections the drum causes on the structure of the machine.

7 Conclusion and Future Research

We have described the theory and implementation of a discrete mechanics model for deformable bodies. The model attempts to preserve physical principles by starting at the atomic level, and then recursively approximating groups of basic elements into fewer larger elements. We have practicly demonstrated that the model incorporates behaviors such as motion, collision and deformation, and theoretically shown behaviors such as fracture and fusing. We have presented two main applications in the conceptual design domain for this model: interactive shape modeling/geometric modeling (virtual claying) and rapid analysis. To support the claim that the model is suited for rapid evaluation, we have presented an algorithm analysis, and shown that the most expensive algorithm is collision-detection, and that algorithms exists which are provably $O(n \lg n + m)$, where n is the number of shapes considered, and m the number of shape pairs which are "very close". Fundamentally, our system can be represented as a mass-spring system, and thus shares many of the weaknesses and strengths of such a system. For example, bodies made of very rigid materials require a very fine time discretization, with long computation time as a result.

Possible future research directions are:

(1) Plasticity

Currently, we incorporate elasticity and viscosity as material phenomena. While we have an artificial model of plasticity, a physics based plasticity model could have large benefits. There has been work done in this area which

may be directly applicable to this model [Terzopoulos, Fleischer, 1988].

(2) Experimental verification

Before the model can be practically used, we need to perform experimental verification of the modeled phenomena.

(3) Material mapping

Related to experimental verification, we need to be able to map real world materials to parameters in the model, so we can translate a description of a real world system of bodies into the model.

(4) Dynamic resolution change

Our theory is based on a recursive resolution reduction. If we can show that a given collection of low-resolution elements sufficiently approximates a given collection of higher-resolution elements, we can dynamically replace the two representations at will. While this is true in our theory, we have not shown how it practically can be done, for example how the parameters of elements are dependent on resolution.

(5) Interfacing with rigid-body representations

If we can represent bodies using the standard rigid-body formulation, we can at least partly overcome the computational expense of simulating rigid materials. This should be fairly straightforward, but requires testing, and perhaps examination of additional possibilities of such a hybrid model.

Acknowledgments

This research has been performed as part of the Integrated Concept Advancement (ICA) project, at the Delft University of Technology.

References

[Andersson, Sellgren, 1998] Andersson, K., Sellgren, U., ”Modeling and Simulation of Physical Behavior of Complex Products”, Proceedings of Produktmodeller -98, Linköping, 10/11/1998 (1998).

[Baraff, Witkin, 1992] Baraff, D., Witkin, A., “Dynamic Simulation of Non-Penetrating Flexible Bodies”. In *Computer Graphics* (Proc. SIGGRAPH) volume 26, pp. 303-308 (1992).

[Berkley, et. al., 1999] Berkley, J., Weghorts, S., Gladstone, H., “Real-Time Finite Element modeling with Haptic Support”. *Proceedings of the ASME Design Engineering Technical Conferences* (1999).

[Bullinger, et. al., 1999] Bullinger, H., Breining, R., Bauer, W. “Virtual Prototyping - State of the Art in Product Design” In: Proceedings of the 26th International Conference on Computers and Industrial Engineering. Melbourne, pp 103-107 (1999).

[Chen, et. al., 1998] “Physically-based Animation of Volumetric Objects”. Proceeding of *IEEE Computer Animation ’98*, pp. 154-160 (1998).

[Cohen, et. al. 1995] Cohen, J., Lin, M., Manocha, D., Ponamgi, M. “I-COLLIDE: An Interactive and Exact Collision Detec-tion System for Large-Scale Environments”. In Proc. of ACM Interactive 3D Graphics Conference, pages 189-196 (1995).

[Cormen, Leiserson, Rivest, 1998] Cormen, H., Leiserson, C., Rivest, R., *Introduction to Algorithms*. MIT Press (1998).

[Delsinger, et. al. 2000] Deisinger, J., Blach, R., Wesche, G., Breining, R., Simon, A. “Towards Immersive Modeling - Challenges and Recommendations: A Workshop Analyzing the Needs of Designers”. In proceedings of 6th Eurographics Workshop on Virtual Environments (2000).

[Fuchs, et. al. 1980] Fuchs, H., Z. M. Kedem, and B. F. Naylor, ”On Visible Surface Generation by A Priori Tree Structures”. *SIGGRAPH* 80, pp. 124-133 (1980).

[Ganovelli, et. al., 2000] Ganovelli, F., Cignoni, P., Montani, C., Scopigno, R., “A Multiresolution Model for Soft Objects Supporting Interactive Cuts and Lacerations”. *Eurographics 2000*, Volume 19, Number 3 (2000).

[Goldstein, 1950] Goldstein, H., *Classical Mechanics*, Addison-Wesley, Reading, MA. (1950).

[Grandin, 1986] Grandin Jr., H., *Fundamentals of the Finite Element Method*. Macmillan Publishing (1986).

[Heath, 1997] Heath, M., *Sceintific Computing*. The McGraw-Hill Companies, Inc. (1997).

[Hollerbach, et. al., 1997] Hollerbach, J. M., Cohen E., Thompson, W., Freier, R., Johnson, D., Nahvi, A., Nelson, D., Thompson, T. V. “Haptic Interfacing for Virtual Prototyping of Mechanical CAD Designs”. *Proceedings of the ASME Design Engineering Technical Conferences* (1997).

[Horváth, et. al., 1998] Horváth, I., Kuczogi, G., Staub, G. “Spatial Behavioural Simulation of Mechanical Objects”. *Proceedings of TMCE ’98 Tools and Methods of Concurrent Engineering Symposium*, pp. 221-223 (1998).

[Hunter, Pullan, 1997] Hunter, P. J., Pullan, A. J., “FEM/BEM notes” (1997). `http://www.esc.auckland.ac.nz/Academic/Texts/FEM-BEM-notes.html`

[ICA group web site, 2000] ICA group web site. `http://www.io.tudelft.nl/research/ica/`

[James, Pai, 1999] James, D., Pai, D., ”Accurate Real Time Deformable Objects”. *SIGGRAPH* 99 (1999).

[Jansson, Vergeest, 2000] Jansson, J., Vergeest, J. S. M., “A General Mechanics Model for Systems of Deformable Solids”. Proceedings International Symposium On Tools and Methods for Concurrent Engineering 2000 (2000).

[Jansson, Horváth, Vergeest, 2000] Jansson, J., Horváth, I., Vergeest, J. S. M., ”Implementation and Analysis of a Mechanics Simulation Module for Use in a Conceptual Design System”. Proceedings ASME Design Engineering Technical Conferences 2000 (2000).

[Kang, Kak, 1996] Kang, H., Kak, A., ”Deforming Virtual Objects Interactively in Accordance with an Elastic Model”. *Computer Aided Design* (1996).

[Keller, et. al., 1993] Keller, H., Stolz, H., Ziegler, A., Bräunl, T., “Virtual Mechanics - Simulation and Animation of Rigid Body Systems”. *Computer Science Report* No. 8/93 (1993). `http://www.ee.uwa.edu.au/~braunl/aero/ftp/docu.english.ps.gz`

[Kleppner, Kolenkow, 1978] Kleppner, D., Kolenkow, R. “An Introduction to Mechanics”, pp. 91. McGraw-Hill (1978).

[Pedersen, 2000] Pedersen, P., “Elasticity - Anisotropy - Laminates” (2000). `http://www.fam.dtu.dk/html/pp.html`

[Ping, Nanxin, 2000] Ping, G., Nanxin, W., ”DOE Study in Building Surrogate Models For Complex Systems”. Proceedings ASME Design Engineering Technical Conferences 2000 (2000).

[Popov, 1999] Popov, E.., *Engineering Mechanics of Solids*. Prentice-Hall, Inc. (1999).

[Provot, 1995] Provot, X. “Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior”. Proceedings *Graphics Interface* ’95, pp. 147-154 (1995).

[Sederberg, Parry, 1986] Sederberg, T., Parry, S., “Free-Form Deformation of Solid Primitives,” Computer Graphics, August, 1986, 151–160. (1986)

[Szeliski, Tonnesen, 1992] Szeliski, R., Tonnesen, D., ”Surface Modeling with Oriented Particle Systems” Computer Graphics, Vol 26., No. 2, July 1992.

[Terzopoulos, et. al., 1987] Terzopoulous, D., Platt, J., Barr, A., Fleischer, K. ”Elastically Deformable Models”. SIGGRAPH’87 pp. 205-214 (1987).

[Terzopoulos, Fleischer, 1988] Terzopoulous, D., Fleischer, K. ”Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture”. SIGGRAPH’88 pp. 269-278 (1988).

[Wiegers, et. al., 1999] Wiegers, T., Horváth, I., Vergeest, J. S. M., Opiyo, E. Z., Kuczogi, G., “Requirements for highly interactive system interfaces to support conceptual design”. *CIRP99* (1999).