FROM SOLIDS TO PARTICLE-BASED MODELS

Monica Bordegoni, Claudio Cilloni, Franco De Angelis
Dept. of Industrial Engineering
University of Parma
I-43100 Parma, Italy
[mb,cilloni,fda]@ied.unipr.it

Abstract This paper addresses the issue of generating mathematical models of flexible objects suitable for NRM simulation. These models should contain proper information not only on the objects geometry, but also on the physical properties of the material they are made of. This paper mainly addresses the geometrical issues of generating particle-based models of flexible objects, but basic methods for determining the material characterization in some special cases are also discussed. The ongoing research work performed by the authors and the related results are presented.

Keywords: Non Rigid Materials simulation, particle models

1. Introduction

There is an increasing market demand for the simulation of non-rigid objects. In industrial sectors where traditionally CAD systems are used (i.e., automotive, aeronautics, ...) there are situations where the simulation of non-rigid parts is required and not available (for example, the simulation of wires, seats, etc.). There are industrial sectors where CAD systems are not used since they do not support modeling of non-rigid products (i.e., textile, fashion, food, ...). Finally there are novel sectors like the medical area, where the simulation of non-rigid bodies would support and improve several critical tasks, like training of procedures and pre-op simulation.

The simulation of objects behaviour requires the creation of a model. Traditional geometric modeling is a mature technology but is not appropriate for non-rigid objects simulation because the generated models lack the kind of information which is needed by a non-rigid material simulator. Current solid models mainly carry information on the geometrical and topological aspects of the model's "outside" (its boundary), plus optional information of the overall volume and mass, assuming the model is rigid and made of a homogeneous and isotropic material.

Instead, a non-rigid material simulator needs detailed information on the physical properties of the model's "inside", which may also be made of a non-homogeneous and anisotropic material, in order to compute its behaviour when it is subject to external forces or deformations. The geometry of a flexible object is not constant, but rather a function of time:

$$\text{shape} = f(t) \tag{1}$$

The actual type of information needed to represent the model depends on the kind of simulation which has to be performed on it. Currently, there is not a common agreement on a method for performing non-rigid materials simulation. Various approaches and techniques are experimented by the researchers in this area. Most of these simulation techniques fall either into the Finite Element Methods (FEM) or into the Particle-Based methods (also known as spring-mass methods or as spring-mass-damper methods) [Cugini et al., 1999]. Our research group has traditionally developed and used a particle-based modeling technique since it offers a simple and general method for modeling objects [Denti et al., 1995] [SIGGRAPH, 1999].

A number of research works based on particle-based simulations have been presented in the last few years. Most, if not all, of them use their own representation of the model, which is often very specific to the application. Especialy in the area of real-time simulations, where performances are the main issue, there is a strict coupling between the simulator and the data structures it uses, aimed at maximizing speed. Therefore, there are blood vessels simulators, liver simulators, and brain simulators [Westwood, 1999], cloth simulators [Volino and Thalmann, 1997] [Breen et al., 1993], pipe simulators, etc. Each simulator has its own specific model, represented in some proper way, and cannot simulate anything else. What we found is the lack of a general method for describing the model of a flexible object.

Recently, the focus of our research has moved further addressing the issue of real-time simulation of non rigid-objects behaviour. Real-time simulation is a requirement in some applications, for example, in those applications supporting haptic interaction with non-rigid objects [Bordegoni and De Angelis, 1999]. Today, a real-time simulation is computationally too expensive to be performed on a complete phisically-based model, hence several real-time applications implement a physically-plausible material behaviour rather than a physically-realistic one.

This paper presents the research work we have developed addressing the issue of generating particle-based models of flexible objects

2. Issues

First of all, let's see how a particle model is made and the kind of information it carries.

The particle-based model technique is based on a discretization of an object volume into a set of point masses (particles) connected by links (often modelled as springs and dampers, i.e., attracting and repelling forces) [Witkin, 1995]. Springs and dampers model the interaction laws among the particles determining the dynamic behavior of the object material. The particle positions vary according to the forces applied. Summaryzing, the fundamental elements used in particle-based modeling are particles, masses and links (Figure 1).

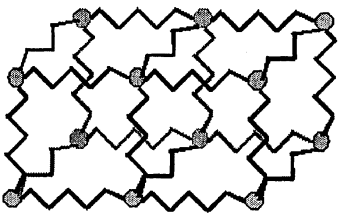

Figure 1. Particle-based model

The first step to perform for modeling a non-rigid object is the definition of the grid of particles approximating the object volume. Once the volume has been discretized, the following step is the definition of the links between the particles. The main issue targeted by our work is the following: given an object of an arbitrary shape, how to discretize it into a particle-based model in order to allow physically-based simulation to be run on it? The aim is to find a discretizing method for placing particles and setting links that adapts to the object initial shape and material.

Therefore, the main issues involved are geometrical ones (shape) and physical ones (constraints imposed on how the shape vary over space and time).

2.1. Geometrical issues

What we mean by "Geometrical issues" is how to place the particles in space, and how to connect them by links. This depends both on the geometry of the object to be discretized, and on the requested level of discretization. Often, this task is executed by users placing particles after particles in space. Of course, this task is tediuous and requires deep knowledge of particle-based modeling and related parameters.

The "Geometrical issue" to address relates to mesh generation. Several methods and algorithms have been proposed in literature for solving the problem

of mesh generation [Ho-Le, 1988]. Most of these methods take solid models in input and compute finite element meshes, using hexahedral for the discretization [Schneiders, 1999]. The result is a lattice of volumetric elements connected at discrete node points. Works on discretization of objects into particles are less common, and mainly concern surface discretization [Shimada et al., 1997].

Our method aims at producing any type of models. A homogeneous and isotropic volume is discretized adopting a regular segmentation using cubic cells (voxels). Conversely, non-homogeneous and anisotropic objects may be characterized by non-regular segmentations.

If the non-rigid object to be simulated is a box, it is straightforward to derive a particle-based model from it. It is sufficient to divide each side of the box into segments (not necessarily equal), and consider the 3D voxel grid implied by such division: each voxel vertex is the position of a particle, and each voxel edge or diagonal is a link. Real non-rigid objects can have any geometry. Geometries like primitives or specific objects can be discretized as well, developing ad-hoc subdivision strategies for each case. But how to deal with objects having an arbitrary shape?

2.2. Physical issues

What we mean by "Physical issues" is which values to assign to the various attributes of particles and links (i.e., masses, elastic constants, etc.). This depends on the physical properties of the object to be simulated, and on the way its geometry has been discretized.

By now, we do not know about any general method which is able to assign the proper values to the physical attributes of particles and links given as input an arbitrary discretization of a 3D object having an arbitrary shape and possibly made of a non-homogeneous and anisotropic material. It is also unknown a general and "abstract" way for describing the object properties to be mapped into the discretized model.

3. Geometric modeling

Since we do not know how to deal with the physical properties of an arbitrary discretized object, our approach to the geometrical issue is to discretize the objects over a regular voxel grid, which makes it possible to assign the physical attribute values in case the objects are made of homogeneous and isotropic materials.

Given these limitations, the work we have done so far deals with objects of arbitrary shape. The object, in the form of a solid model, is imported into a modeler and then processed for generating the particle model. We are exploring different methods for performing the processing, which are described in the following section 3.1.

The work is based on the commercial solid modeler ACIS [Spatial, 2000], but the application has been thought to be "modeler-independent" so a different modeler might be used.

3.1. Algorithms

The common part of all algorithms we have developed is that the model is intersected against a regular grid of voxels (one at a time). Each time a voxel is intersected with the model, there are three possible cases:

1 The intersection is null. This means the voxel does not belong to the model, and can be ignored.

2 The intersection is coincident to the voxel. This means the voxel is internal to the model. Like in the box case, a particle is placed at each vertex of the voxel, and a link is set for each edge or diagonal.

3 The intersection is not null, but differs from the voxel. This is what we call "the broken voxels" and means that the voxel is across the model's boundary.

The critical part of the work is how to deal with the broken voxels. As the broken voxels have in general more vertices and edges than a complete voxel, geometrical issues concern which vertices should be used as particles and how to connect them.

The resolution of the physical model, that is the number of particles and links, should be minimized in order to reduce the computational cost during simulation time. Because of real-time constraints, we keep the number of relations between particles low. In fact, each relation implies a set of equations to be solved. Increasing the number of equations increases the computational time. In case of haptic applications, a processing delay would provoke a discrepancy between what the user expects of perceiving (visual and haptic) and what she really perceives.

Besides, because the distribution of particles and links is more "dense" on the model's boundary than in its internal part, it is difficult assigning the values of the physical attributes. As already stated, determining and assigning the physical parameters to an arbitrary discretization is an open issue.

Some algorithms we are experimenting for treating the broken voxels are the following ones:

- the broken voxels are approximated for defect (ignored) or approximated for eccess (considered as they were full), depending on some set criteria (e.g. a treshold on the occupancy ratio).
- the broken voxels are ignored or approximated for their bounding box, depending on some set criteria.

- the broken voxels' boundary is faceted by the modeler. In this case a huge number of particles and links is generated, which is undesirable.
- the broken voxels are approximated by a convex polyhedron having a maximum number of faces and vertices known in advance and lower than in the previos case.

In the following we present each of these algorithms.

3.1.1 approximation by voxel. This simple algorithm just discards a broken voxel, or approximates it for the complete voxel that originated it, depending on a criteria. The adopted criteria is based on computing the ratio between the broken voxel's volume and the complete voxel's volume. If this ratio exceeds a given treshold the broken voxel is approximated for the full voxel, otherwise it is discarded. As special cases, if the treshold is set to 0.0 every broken voxel is kept, while if the treshold is set to 1.0 every broken voxel is discarded. Figure 2 shows the different output of this algoritm when is run on the same object, same voxel grid but using different values for the treshold.

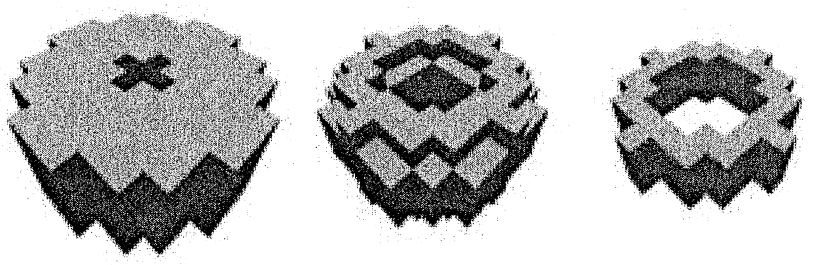

Figure 2. Voxel-level approximation of a pipe slice using a threshold of 0.0, 0.5 and 1.0

3.1.2 approximation by bounding box. This algorithm is a refinement of the previous one. The bounding box of the broken voxel is considered and its occupancy ratio is computed; if the ratio exceeds a treshold the broken voxel is approximated for its bounding box, otherwise it is discarded. Figure 3 shows the different output of this algorithm when run on the same object, same voxel grid but using different values of the treshold.

This method always produce better results than the previous one relatively to the overall approximation of the original object's shape (see also Section 3.2), but arises issues in assigning physical properties to particles and links, as the approximated voxels differ from the complete ones.

3.1.3 approximation by a mesh of polygons. Another solution is to keep the broken voxel "as is". The broken voxel's boundary is approximated for

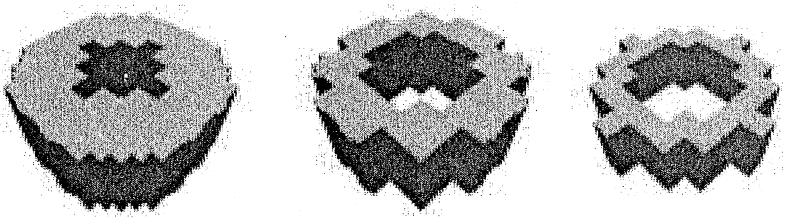

Figure 3. Bound box-level approximation of a pipe slice using a treshold of 0.0, 0.5 and 1.0

a triangle mesh by the modeler; a particle is placed at each triangle vertex and a link is created for each triangle side. Optional links can be added to connect particles belonging to different triangles.

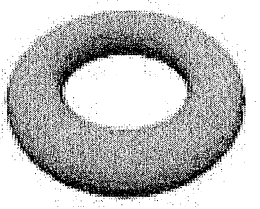

Figure 4. "Exact" approximation of a torus for a triangle mesh

As it can be seen in Figure 4, the output of this algorithm is graphically nice, but depending on the broken voxels's geometry and the way the triangle mesh is generated it can produce a huge number of particles and links, making the simulation of the output model computationally too expensive. Furthermore, the "random" distribution of particles and links in the broken voxels arises open issues for the physical properties assignement.

A refinement of this method attempts to reduce the number of polygons in the mesh. The bounding box of the broken voxel is determined, and each vertex of the mesh is classified depending on its position in the bounding box. The vertices of the mesh could lie on a vertex of the bound box, on an edge, on a face or be internal: the internal ones are discarded and the remaining ones are re-linked.

This greatly reduces the number of vertices and links, but still produces a "random" mesh leaving the issue of assigning physical properties open.

3.1.4 approximation by a convex polyhedron. The difficulties encountered in the previous algorithm led to search for a different solution. What

we actually want to do is to approximate the broken voxel with a polyhedron whose vertices always lie on the bounding box edges, as shown in Figure 5.

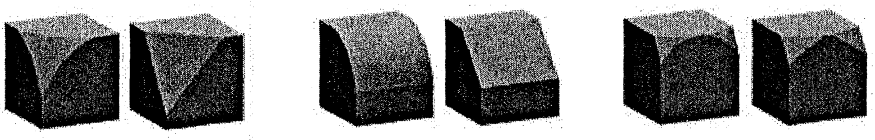

Figure 5. Samples of broken voxels and their expected approximation

This requirement not only reduces the number of generated vertices (i.e., particles), but also allows us to establish some rules for connecting them by links and to associate physical properties to them.

The algorithm developed to accomplish this result is based on the primitive shown in Figure 6.

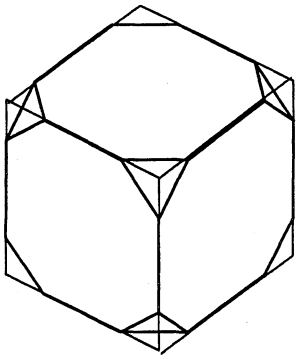

Figure 6. The polyhedral primitive used

This primitive is a convex polyhedron made of 24 vertices, each of them having one degree of freedom allowing it to move along one edge of a box. As a special case, this primitive can generate the box itself, as well as other convex shapes contained in the box. Examples are shown in Figure 7.

The algorithm approximates each broken voxel using this primitive. At first, the primitive is configured so that it is coincident to the broken voxel's bounding box (three vertices of the primitive lie on each vertex of the box). Then, for each vertex of the box a search process is performed, aimed at finding a better approximation by sliding the three primitive vertices associated with that box

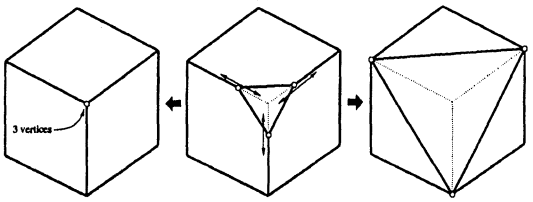

Figure 7. Sample shapes generated by the primitive

vertex, along the box edge corresponding to their given degree of freedom. The main steps of this search process follow.

- Each one of the three vertices is moved away from its initial position, along its associated box edge, until it gets in touch with the broken voxel (the search is successful) or it reaches the opposite end of the edge (the search fails).
- If the search is successful for all the three vertices, the three vertices define a triangle of the approximating polyhedron and a particle is placed on each vertex of the triangle.
- If the search fails for one or more vertices, a number of cases requiring some special handling are generated. Not all the cases are implemented yet: when one of these special cases is encountered, if its handling is implemented it is treated accordingly, otherwise the voxel is discarded (approximated for defect). Figure 8 shows the output of this algorithm run on a sample object.

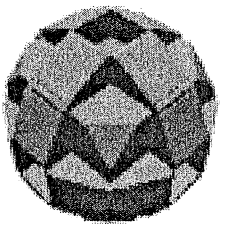

Figure 8. Approximation of a sphere by the polyhedral primitive algorithm

Whichever algorithm was chosen, at the end of the process a reduction pass is performed, in order to discard redundant (coincident) particles and links, preserve the model integrity and speed up its computation.

This pass must also deal with the physical properties associated with the particles and their links, to avoid information loss and changes in the object behaviour: for example when discarding a particle coincident with another one, its associated mass has to be "transferred" into the remaining one to preserve the total object mass.

3.2. Result evaluation

In order to show how the various algorithms work, we need to run them on one or more common test cases and compare their output. In order to properly compare the various algorithms, a purely visual evaluation of their output is not enough and too subjective; a numerical method of error evaluation is needed.

3.2.1 chosen criteria. Comparing the volumes of the original model and the discretized model may seem a good idea, but it is not: even two totally different models may happen to have the same volume.

Computing the following formula:

$$\frac{\sum_{i=1}^{N} |\text{occupancy(orig.model,voxel}_i) - \text{occupancy(disc.model,voxel}_i)|}{\text{original model volume}} \quad (2)$$

(where N is the number of voxels) seems to be a better idea: in this case, the computed value (the "error") is zero if, and only if, the two models are coincident.

3.2.2 algorithm comparison. In order to show how the various algorithms behave, a few objects were considered and a set of tests was performed for each object. A "set of tests" corresponds to executing all the algorithms on the same object, eventually changing algorithm parameters but without changing the resolution of the voxel grid, and computing the resulting "error" as in formula (2). In the following Table 1, each row corresponds to a set of tests run on the same object.

Table 1. Errors computed for some objects. Columns (algorithms): v voxel based, b bounding box, polyh polyhedral primitive, t treshold value.

	v,t0.25	*v,t0.5*	*v,t0.75*	*b,t0.25*	*b,t0.5*	*b,t0.75*	*polyh*
pipe	*66.3%*	*52.6%*	*73.15%*	*16.5%*	*12.5%*	*22.2%*	*6.8%*
sphere	*41.0%*	*30.3%*	*46.2%*	*30.1%*	*24.9%*	*34.9%*	*20.1%*

The error for the triangle mesh algorithm is not computed but can be considered null as the triangle mesh is self-generated by the modeler (hence it basically corresponds to the broken voxel itself).

As it could be expected, given a treshold the bound box-approximation algorithm always produces better results than the voxel-approximation algorithm, and a treshold of 0.5 is always the best choice for both. However, whichever treshold is chosen, the polyhedral primitive algorithm gives better results.

3.2.3 resolution dependence. Another group of tests was performed to show how the resolution of the voxel grid influences the results. Here, all the tests were performed on the same object (a sphere), but using a different resolution of the voxel grid for each set of tests. Table 2 sums up the results.

Table 2. Error, number of particles and number of links as a function of the voxel grid resolution. The voxels are cubical and their side (the grid step) is expressed as a fraction of the sphere diameter.

grid step	*voxel t0.5*	*bbox t0.5*	*polyh*	*polyh particles*	*polyh links*	*mesh particles*	*mesh links*
0.3	*34.4%*	*28.8%*	*32.9%*	*128*	*472*	*610*	*1940*
0.2	*31.0%*	*18.9%*	*35.2%*	*176*	*1204*	*562*	*1868*
0.15	*22.9%*	*11.6%*	*15.5%*	*540*	*3904*	*858*	*3408*
0.1	*14.5%*	*10.0%*	*10.9%*	*1132*	*9808*	*1458*	*7904*

As expected, the volumetric error reduces as the resolution increases, but the geometrical issue is not the only one to be considered. The four rightmost columns of Table 2 report the number of particles and links generated by the polyhedron algorithm and by the triangle mesh algorithm: these numbers can't exceed the computational capacity of the employed simulator.

That's why our current work aims to refine the "polyhedron" algorithm, rather than just increasing the resolution of the discretization: the objective is to get a reasonably nice result, while keeping the resolution as low as possible.

4. Physical parameters modeling

Up to now, we do not know of any method for automatically setting physical parameters, given the physical properties of the material and an arbitrary discretization of the object. As a first approximation, we apply some simplifications to the model and the physical rules so as to get a simpler and "lighter" model. This can be of use, for example, in our real-time simulator of non-rigid materials integrated with haptic rendering [Bordegoni and De Angelis, 1999] and related applications.

Our current work includes a basic treatment of physical properties at least for isotropic materials discretized over a regular grid. In this case, it is safe to state that all the particles have the same mass, and all the "similar" links (i.e. those having same length and orientation in space) have the same physical

properties. Thus, if some values for the physical attributes are given, they can be properly associated to particles and links during the discretization phase. Also, if during the algorithm execution some particles or same links of the same kind collapse together, their physical attributes are treated accordingly to determine the physical attributes of the resulting particle or link (e.g. masses of particles are summed together).

In the case of isotropic materials discretized over a regular grid, different methods can be used for determining the physical attributes to be associated to particle and links:

- Interactive adjustment done by an expert user;
- Measure parameters.

Concerning the first method, we are implementing an application for tuning the non-rigid model parameters. The application is integrated with haptic devices [Bordegoni and De Angelis, 1999]. The user can set the parameter values (constant of traction, compression, damping, ...) using a graphical user interface and immediately touch the object model wearing the haptic devices. The model is better and better approximated by adjusting the parameters according to the impression of resemblance between the real object and its corresponding model.

According to the second method, physical parameters can be measured through some experimental tests. For example, fabric mechanical properties can be measured by means of the Kawabata evaluation system [Kawabata, 1980].

With the first method, the validation of the model is immediately done by the user setting the parameters. The second method requires the model to be compared with experimental results.

5. Conclusions and future work

The work done so far is able to produce a reasonable discretization of an arbitrary shape model over an arbitrary resolution, though there is room for improvement of the algorithms.

However, there is only a basic treatment of physical properties, and only isotropic materials discretized over a regular grid can be handled. The future work will target open issues like these:

- how to represent the physical properties of a non rigid, eventually non isotropic, material in a general way?
- how to map the physical properties of an object made of a non rigid material into its arbitrarily discretized model?

References

Bordegoni, M. and De Angelis, F. (1999). Haptic interaction with non-rigid material models. *Convegno ADM on "Design Tools and Methods in Industrial Engineering"*, XI.

Breen, D., House, D. H., and Wozny, M. (1993). A particle based model for simulating the draping behaviour of woven cloth. *Technical Report n. 93011. Rensselaer Design and Manufacturing Institute, Rensselear Polythecnic Institute.*

Cugini, U., Bordegoni, M., Rizzi, C., De Angelis, F., and Prati, M. (Milano 1999). Modelling and haptic interaction with non rigid materials. *Eurographics '99 - State-of-the-Art Reports*, pages 1–20.

Denti, P., Ursino, M., and Rizzi, C. (1995). Simulated behaviour of non rigid materials. *International Journal of CADCAM and Computer Graphics*, 10(1-2).

Ho-Le, K. (1988). Finite element mesh generation methods: a review and classification. *Computer Aided Design*, 20(1).

Kawabata, S. (1980). *The Standardization and Analysis of Hand Evaluation.* The Textile Machinery Society of Japan, Osaka.

Schneiders, R. (1999). Automatic generation of hexahedral finite element meshes. *8th International Meshing Roundtable.*

Shimada, K., Yamada, A., and Itoh, T. (1997). Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. *6th International Meshing Roundtable.*

SIGGRAPH, editor (1999). *Physically Based Modeling.* Course Note n. 36.

Spatial (2000). *The ACIS 3D Modeler.* Spatial Technology, http://www.spatial.com/.

Volino, P. and Thalmann, N. (1997). Developing simulation techniques for an interactive clothing system. *Proceedings of VSMM 97, IEEE Computer Society*, pages 109–118.

Westwood, J., editor (1999). *Medicine Meets Virtual Reality*, number 7 in MMVR. IOS Press.

Witkin, A. (1995). Particle system dynamics. *ACM SIGGRAPH Course Notes*, 34:C1–C12.