

# Introducing Multiresolution Representation in Deformable Object Modeling

Fabio Ganovelli, I.E.I. - C.N.R. \*

Paolo Cignoni, I.E.I. - C.N.R. †

Roberto Scopigno, CNUCE - C.N.R. ‡

## Abstract

The need to simulate the behaviour of biological tissues in virtual environments originated intense research to devise techniques for physically-based modeling of objects whose shape and topology can dynamically evolve. The relevant amount of computation required by the existing techniques and the growing complexity of the scenes to be simulated impose the study of more flexible methods that should be adaptable to both the accuracy required by the simulation and the complexity of the scene. We propose to enhance the flexibility of deformable object models by introducing a multiresolution approach. The reason for proposing the adoption of a multiresolution approach in deformable object modeling is to allow easy adaptation of the granularity of the representation of each object, depending from the forecasted simulation time complexity and the interactive frame rate required. Multiresolution can also be used to adjust data granularity, according to the accuracy needed to preserve the properties of the material and to obtain a sufficiently correct dynamic behaviour of the object. A first design of a multiresolution deformable model is presented in the paper, based on hexahedral cells and a hierarchical decomposition scheme.

## 1 Introduction

Graphics is becoming the main media in many applications, not only to provide simple static visual cues but also to support interactive and immersive representations of real or synthetic 3D environments. Impressive is the recent evolution of methodologies enabling interactive real-time applications, nowadays generally known as “Virtual Reality” applications. Computer Aided Surgery is the field where most remarkably consequences can be appreciated, for example having an environment for off line surgeon’s training. A fundamental requirement for this kind of applications is that the objects involved in a scene should deform under external loads, since that most of them cannot be assumed as rigid. The introduction of this requirement has considerably complicated the management of virtual

scene, because it involves time consuming calculations to compute the physically based shape variations and more complicated solutions for collision detection. The set of methodologies apt to solve these problems are referred as *Soft Tissue Modeling* when their application domain is devoted to the simulation of human organs or, more in general, *Deformable Object Modeling*. Although this field is quite new, a variety of solutions has been proposed and some of them are starting to be used in prototypal systems [5].

Proposed approaches can be summarized on the base of their goal: the *accurate* prediction of tissue repositioning after external interaction (*surgical planning*), or the *approximate real time* emulation of the behaviour of the tissue (*surgical training*). The differences between an approach devoted to real time and one devoted to maximal accuracy reside in: the different precision in the definition of the material properties while formulating the equations governing the deformations, the numerical technique adopted for the resolution of the equations, and the granularity of the object model used. Model granularity can be considered as the number of elements for unity of volume, and obviously it directly affects the frame rate obtainable in a dynamic simulation. This work focuses on this point: the multiresolution approach, i.e. those technique for storing and extracting multiple representations of the same space at a different Level of Detail, are here extended to Deformable Modeling. In this context, the choice of an ideal Level of Detail is not only related to the granularity of the representation but also to the locality or globality of the behaviour and to the required accuracy the same behaviour has to be simulated. The more a representation is coarse (fine), the more its behaviour is approximate (accurate), the faster (slower) is the frame rate. This suggests that multiresolution representation could make the same numerical approach adaptable to different goals and also to different scenes. For example, any system for virtual reality supporting the interactive manipulation of deformable objects and running on a given processing architecture has a limitation in the maximum number  $M$  of cell elements used to describe the objects/scene; we want to design a methodology that, given a model/scene larger than  $M$ , will allow dynamic selection of the resolution of the objects involved in the scene so that the number of cells would not

---

\*Email: ganovelli@iei.pi.cnr.it

†Email: cignoni@iei.pi.cnr.it

‡Email: r.scopigno@cnuce.cnr.it

exceed  $M$ .

The paper is organized as follows. In Section 2 we introduce some basic concepts from the existing literature on deformable object modeling, the mass spring model, and some problem related to the application of the mass spring model to deformable tissue modeling. In Section 3 a general multiresolution framework is introduced in its native context and, very briefly, its use in visualization is presented. Section 4 is the core of the work: we introduce a multiresolution approach to Deformable Object Modeling, based on a regular hierarchical decomposition of the domain in hexahedral cells, and where the material is supposed to be linearly elastic. In Section 5 we give some concluding remarks and some detail on the planned experimentation of this on-going project.

## 2 Previous works

Main issues in the simulation of deformable objects are how to represent both the shape and the physical parameters of the object and how to drive its dynamic deformation or behaviour. With regard to the first issue, an object can be represented by adopting an analytical or a polyhedral cell-based description. Analytical representations (e.g. parametric or implicit surfaces) give several advantages with respect to polyhedral ones. The main one is that a parametric surface can be deformed by maintaining its smoothness independently from the complexity of the deformation, while a polyhedral representation in general cannot do it without augmenting the decomposition resolution. Secondly, efficiency of inside/outside test for analytical descriptions offers great advantages for handling collision detection and response (see [20]). A model for deformable modeling using implicit surfaces is presented in [4]. Furthermore, analytical or parametric data description is in general very synthetic, and a deformation can simply be specified by moving few control points of the parametric surface.

On the other hand, the specification of *local deformations* and the computation of a physical behaviour is not easy in the case of analytical representations [10]. For example, the Free Form Deformation (FFD) approach [19, 24], is based on the idea of deforming the object by deforming the surrounding space. The latter is divided into regular cells, and any modification of the geometry of the surrounding lattice causes a direct feedback on the object shape. This method suffers of the drawback cited above (how to introduce local deformation), and an extended model has been proposed to partially overcome this deficiency [7]. Even if physically-dependent constraints on the motion of grid points have been introduced for the aim of animation [24], this approach is not commonly used for the simulation of deformable objects characterized by very complex topologies (and/or dynamic topology) and can hardly reproduce the shape/behaviour of an object acquired from the real world (for example the behaviour of a complex human or-

gan).

The approaches proposed for the simulation of deformable objects are mostly based on laws of the Newtonian mechanics. The displacement of each point of an object subjected to a load force  $f$  is governed by the equation:

$$m\ddot{x} + \gamma\dot{x} + \frac{\partial\varepsilon(x)}{\partial x} = f \quad (1)$$

where:  $x$  denotes the position of mass point;  $m$  and  $\gamma$  are respectively the mass and the damping factors; the term  $\frac{\partial\varepsilon(x)}{\partial x}$  is the *variational derivative* of the potential energy  $\varepsilon$  due to the motion of the point (it is a force term). The definition of the energy function is a crucial task; pioneering attempts to give a mathematical model of plasticity, viscoelasticity and fracture have been produced by Terzopoulos et. al. [21] and Terzopoulos and Fleischer [23]. In these works the points on the object are defined as the sum of a reference component so that the rigid body motion and the deformation are separated.

### The FEM model

The most widely used, accurate and expensive way to model the deformation of tissues is the *Finite Element Method (FEM)*. This method solves the static equilibrium problem: *internal energy = work done by loads*. Such equation, initially defined in the continuum, can be expressed in matrix form as  $Kx = f$ , where the matrix  $K$  is called *stiffness matrix*, by means of a partition of the domain in finite elements and of the definition of opportune *interpolation functions*. The latter describe the internal of each element as a function of the nodes defining the element. FEM has been used, for example, in the context of planning cranio-facial surgery, by modeling the skin surface of the face with prismatic elements [13] or as a triangular surface attached to the underlying bone by means of one-dimensional linear elastic elements (*springs*) [25]. FEM-based tetrahedral meshes are also used in [18] and in [6].

The works cited above do not pursue the goal of *real-time* simulations, but are oriented to obtain a very *accurate* deformation, which is obviously one of the most important aspects in surgery planning. An attempt to speed up FEM computation has been recently proposed [3] and is based on the *condensation* of the linear system, i.e. it leaves implicit the solution for the nodes internal to the volume (which are generally not visualized) and makes explicit only the ones on the surface; in this manner, a smaller but more dense matrix is obtained, and its inverse can be computed. This approach allows a faster calculation (real-time performances for small meshes of around 250 elements), but it absolutely prevents the possibility to introduce any kind of dynamic topological change in the mesh.

Although new solution techniques and the improvement of hardware performance could speed up FEM computations, the growing complexity of the scenes to be managed makes it reasonable that the gap between the computational cost of this method and real time

frame rate will persist. In other words, it makes sense to investigate less accurate but more computationally efficient techniques.

### The Mass Spring model

The idea of introducing constraints between the mass points that describe the object belongs to many of the existing approaches. These constraints, generally called *springs*, are often used to model muscle actions [26, 22, 15] or to constrain a skin layer to lie on the corresponding skull section. Generally speaking, the mass spring model is often used together with other models, for example to model 1D or 2D entities in a 3D context (see some examples in the context of cloth simulation [1, 2, 16]) rather than to represent directly a 3D object.

With the term Mass Spring System we intend a system formed by a set of *mass points* and a set of constraints between couple of points; each point is subjected to forces due to the status of the springs connected to it and to the potential external forces (gravity, user interaction etc...). The position of the mass points is calculated at each instant by integrating the equation of motion.

It has to be taken into account that the system of equations arising from the previous definition cannot be simply integrated trough the time, because points of unwanted local minima could be generated such that the simulation becomes unrealistic. For clarity, we propose a typical example coming from cloth simulation applications: a rectangular plate is represented in Figure 1.a, with six springs connecting four mass points. It is quite obvious that we want to preserve the topology of this element during the simulation. Suppose that the springs are in their rest state and that a force directed towards the internal of the rectangular plate is applied to the mass point *C*. If we use the Euler Method to solve the differential equation of the motion, at the first time step we could move the point *C* so that the element is no more a quadrilateral; moreover, unloading the force *f*, the mass spring system will converge to a configuration similar to the one in Figure 1.b and we get in an unwanted local minima. This kind of behaviour, that we indicate with the term *cell reversion*, is due to the fact that the mass spring model only consists of a set of relations between mass points and there is no idea of the cell or the space occupied. Note that this phenomenon depends on the magnitude of the forces applied; in other words, for any value of stiffness and mass, and for any value of the original distance between mass points, it exist a force great enough to cause a cell reversion. This problem can be avoided if the parameters value (stiffness, lengths, masses) are "well proportioned" with the order of magnitude of the forces; conversely, it has to be taken into account if we want to guarantee maximal freedom of interaction. We can introduce constraints to the minimal dimension of the cells to preserve their structure (see [12]) and/or we can use either more accurate integration methods or smaller time steps, but the latter strategy easily causes a slow down of the computation.

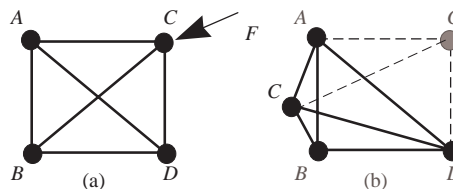


Figure 1: An example of cell reversion.

If we observe that for each force *f* there is a cell of sufficient size such that it cannot be reverted by *f*, then emerges one of the advantages of a multiresolution deformable model: known the force acting on the object, we can choose a representation such that no cell is reversed by the force.

## 3 Multiresolution in Visualization

The design of *multiresolution* or *level of detail* (LOD) approaches for modeling and visualizing data has been a very active field of research in the last few years. The concept of *LOD* or *multiresolution* is generically related to the possibility of using different representations of a geometric object (a surface, a volume, an image, etc.), having different levels of accuracy and complexity. More in detail, a multiresolution mesh is a model that can provide a high number (virtually, a continuous range) of meshes representing a single object at different resolutions.

A general framework has been recently proposed, the *Multi-Triangulation (MT)* [9, 8], which encompasses most of the approaches proposed in literature. This framework is based on the following considerations:

- any multiresolution mesh can be built by means of local operations that progressively modify an initial mesh through either *refinement* or *simplification*;
- local modifications can be arranged into a partial order according to their dependencies.

Let us consider in the following only the approaches based on simplification. Central to the MT framework is the concept of mesh *fragment*, i.e. a [generally small] portion of the mesh, which is the object trasformed by the single *atomic action*. Each simplification atomic action is applied to a given fragment, and will only modify this mesh sub-area (*locality* of the action) by producing a new fragment that replaces the old one [17].

Therefore, central to a given MT framework implementation are:

- the choice of the *legal atomic actions* that will be used in simplification; they have to be local and will determine also the structure of the fragments;
- the overall simplification process can be represented through a DAG, that stores all the fragments involved

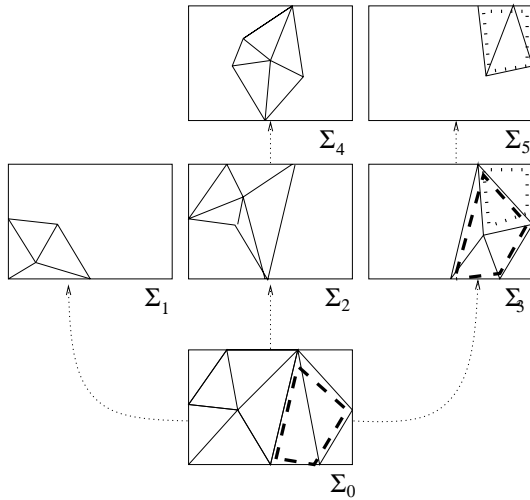


Figure 2: An example of a simple MT model, built on a 2D simplicial decomposition by adopting a simplification approach based on vertex removal.

in the atomic simplification actions (DAG nodes), and the dependencies between fragments (DAG arcs); the DAG has its root at the initial mesh, and each arc connects pair of nodes such that the start node of the arc has some cells “covered” by the fragment corresponding to the end node; an example is in Figure 2.

Obviously, the resulting DAG can be produced as a by-product of many different simplification approaches, and can be further used to extract meshes whose resolution can be either *uniform* over the whole domain represented or *variable*, i.e. gradually changing over different zones of the domain represented.

It has been shown that the extraction of whichever level of detail from an MT model corresponds to the selection of a *cut* on the DAG [9]. The corresponding mesh (or decomposition) will be composed by all of the cells which correspond to the end nodes of the arcs intersected by the cut (see Figure 2).

Some desirable properties of a multiresolution model are: *space efficiency*, i.e. the size of the model must not be considerably higher than the size of the mesh at the highest resolution it can provide; *query processing efficiency*, i.e. it must be possible to extract a mesh from the model at a given resolution in short [real] time; *quality of the query results*, e.g. continuity should be guaranteed on the extracted meshes and transition between different detail level should be as smooth as possible.

Most of the approaches proposed are based on *simplicial* cell decomposition, which informally means either triangle-based surfaces or tetrahedral-based volumes (an example is presented in Figure 2). This is due to some properties of simplicial decompositions: continuity is often easily guaranteed, the determination of effective atomic action is easy, the quality of the results (in terms of geometric accuracy) is often very high.

In the next section we will give some justifications to our choice of selecting a hexahedral-based cell decomposition and a simpler hierarchical dependency scheme between fragments. The overall framework underlying our work remains the MT model, but to reduce the complexity of the design and implementation of a multiresolution Mass-Spring model we decided to consider at first only hierarchical hexahedra-based multiresolution decompositions. Further research is needed to generalize what we propose in the following sections to manage generalized MT models.

## 4 Extending the multiresolution approach to Deformable Object Modeling

Introducing a multiresolution approach in the context of Deformable Object simulation involves three aspects:

1. **Choice of a representation scheme.** Assuming a polyhedral decomposition approach, one of the central point is to choose the basic cell decomposition model (simplicial, hexahedral or a more general cell complex); as we will describe in the following, a decomposition rule based on *hexahedral cells* has been adopted.
2. **Definition of the criteria for simplification and construction of the multiresolution model.** This directly affects the shape of the fragments and the DAG resulting from the simplification. We choose to simplify fragments corresponding to a regular partition of the space in octants, that originates a simple *octree*-shaped dependency graph. Rules for the determination of the physical parameters of the simplified cells have been designed.
3. **Definition of the strategy for extracting a representation of the soft object at a resolution appropriate to the external forces/interactions.** Resolution used in data representation does not depend on view-dependent parameters, but on the interaction of external agents on the soft objects, that is on the field of forces applied to the object.

### 4.1 Initial representation of the object

As we put in evidence in Section 2, a mass spring system does not include the concept of *cell* in its definition, while we need it to describe the space. The simple relations between points in the space given by the springs are not enough neither for efficient visualization nor for efficient simplification and multiresolution modeling.

Our initial mass spring system is derived from a voxelized representation of the domain, simply by introducing a mass point for each voxel vertex and a spring (or,

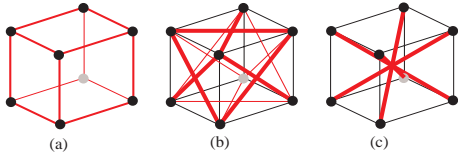


Figure 3: A cell: axis aligned springs (a), face shear springs (b) and body shear springs (c).

more generally, a constraint) between adjacent vertices. These axis-aligned springs represent the resistance to axis directed stress. To include also shear stress, we have to add further springs between the opposite vertices on each voxels face, and for each opposite vertex on voxel internal diagonals. Figure 3 shows the generic hexahedral cell.

## 4.2 Octree-based simplification

An *octree* (*OT*) is a data structure based on a regular, recursive spatial subdivision in octants [14]. The root of the octree corresponds to a hexahedral domain involving the whole object and the eight sons of a node correspond to the eight equal sized partitions of the space occupied by the father node. Each node of the octree is thus directly associated to a region of the domain; border cells (e.g. cells that are not completely internal or external *wrt* the target soft object) give an approximate representation of the shape of the corresponding section of the target object, and the approximation quality depends on the level of the node and on the original shape of the target object.

An octree-based hierarchical representation can be simply built on top of a voxel-based mass spring model. The critical part is here how to compute the values for the masses and the springs of the cells corresponding to the internal nodes of the octree. With respect to the general *MT* framework, each internal *OT* node is a fragment; a characteristic of the *OT* is the regular spatial disposition of the fragments, that simplifies data management (e.g. the DAG reduces to a tree) but may led to not-optimal simplified representations. The regular recursive decomposition of the *OT* makes also simple to compute the characteristic parameters of any internal node at level  $i$  on the basis of the corresponding set of leaf cells<sup>1</sup>.

The method designed to determine the characteristic parameters of a given intermediate node of the octree is described in the next subsection.

We plan to investigate in the next future different simplification rules working on hexahedra decomposition, and to evaluate if the complexity introduced (in simplification and in the determination of the simplified cells parameters) will result in better-shaped and, more important, in a more accurate representation of the physical behaviour.

<sup>1</sup>For each internal node, we avoid to compute its physical parameters from the corresponding values nodes of level  $i + 1$  to avoid to accumulate the error introduced from the previous simplification steps.

## 4.3 Simplification of fragments

The two main points are how to determine both the *mass* of the nodes of the simplified fragment  $\gamma$  and its *springs* characteristics.

Analogously to the simplified models used in visualization, an error is associated to every simplified fragment which represent the object at a given accuracy. In the case of the visualization-oriented multiresolution models, the error is a measure either of the geometric distance or/and of the field difference. Conversely, in the context of soft object modeling error has to be intended [and measured] as the difference in term of behaviour of two corresponding fragments under the application of the same forces (major details in Section 4.5).

A simple method is defined in the following to derive the characteristic parameters of a fragment from the characteristic parameters of the corresponding set of cells. In our octree-based model, the *atomic step* of simplification builds the mass spring cell  $\gamma$  associated to an intermediate octree node from the corresponding set of leaf cells  $\Gamma$ . The assumption that simplifies the job of determining the physic behaviour of simplified fragments is that we always have axis-aligned hexahedral fragments. The further constrain characteristic of the octree model (the relation between fragments' cell is always a  $1 : 2^{(n-i)}$ , with  $i$  the level of the node corresponding to  $\gamma$ , and  $n$  the depth of the octree) is not really mandatory, and that permit to forecast easy extension to other hexahedral-based simplification schemes.

In our current model, we just consider purely elastic springs, with an associated maximum and minimum length. In our method, the values of each spring of  $\gamma$  depends on the equally aligned springs of  $\Gamma$ , and each one of the eight masses of  $\gamma$  depends on all the masses of  $\Gamma$ .

### 4.3.1 Masses of $\gamma$

Each mass  $m_{xyz}$ ,  $x, y, z \in [0, 1]$  of  $\gamma$  is obtained as the weighted sum of the masses of  $\Gamma$ , where the weight of a mass  $m_{ijk}$  is the coefficient for the point  $p_{ijk}$  in the convex combination of the points  $P = \{p_{xyz}\}$ ,  $x, y, z \in [0, 1]$ :

$$m_{x,y,z} = \sum_{i,j,k=0}^{u,v,w} ((1-x) \cdot \frac{j}{u} + x \cdot \frac{u-i}{u}) \cdot ((1-y) \cdot \frac{k}{v} + y \cdot \frac{v-j}{v}) \cdot ((1-z) \cdot \frac{k}{w} + z \cdot \frac{w-k}{w}) \cdot m_{ijk}$$

in this way the masses of  $\Gamma$  are lumped to the nodes of the cell  $\gamma$  proportionally to their distance from each nodes.

### 4.3.2 Axis-aligned springs of $\gamma$

We consider in the following the case of the mass-spring models with *axis*-aligned springs only, for the sake of

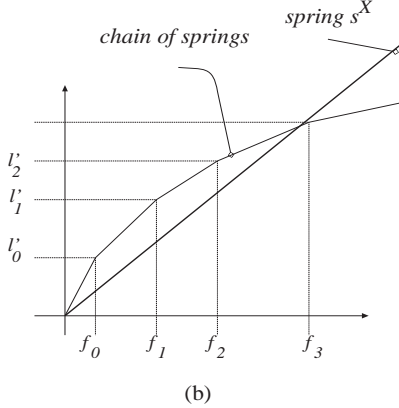
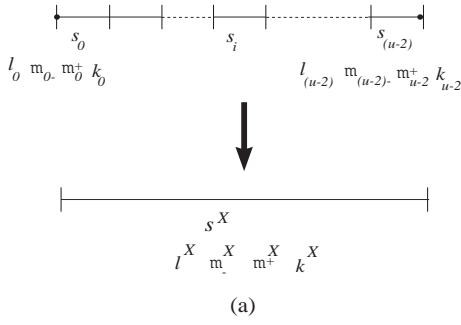


Figure 4: Simplification of a chain of springs in a single one (a); relations force/displacement for the two representations (b).

simplicity. We proceed in two phases<sup>2</sup>:

1. transform each set of springs  $s_{\bar{i}jk}$ ,  $i \in [0, u - 1]$  in a single spring  $s_{jk}^X$ ;
2. transform the set of springs  $s_{jk}^X$ ,  $j \in [0, v]$  and  $k \in [0, w]$  in the four springs  $X$  axis aligned.

This two phases are described in detail in the following.

### Chain of linear springs

A chain of linear springs can be represented, under a certain degree of approximation, with a single linear spring; the approach used to compute the parameters is described below.

For the sake of readability, we isolate the generic set of springs  $s_{\bar{i}jk}$ ,  $i = 0, u - 1$  in Figure 4.a and omit the indices  $jk$  that are fixed for this set. We obtain a chain of spring  $s_0, \dots, s_u$  where each spring  $s_i$  is defined by: a minimum length  $\mu_{i-}$ , a maximal length  $\mu_{i+}$ , a rest length  $l_i$  and a stiffness constant  $k_i$ .

### Minimum length

The minimum length of a chain of springs is trivially obtained as the sum of the minimum lengths of all the springs:

<sup>2</sup>Please note that with  $s_{\bar{i}jk}$  we denote the spring that connects the node  $p_{ijk}$  with the node  $p_{i+1jk}$ , i.e. an X-axes aligned spring.

$$\mu_{i-}^X = \sum_{i=0}^u \mu_{i-}$$

### Constant of stiffness

We define  $f_i = (l_i - \mu_{i-})k_i$  the force necessary to completely contract the spring  $s_i$  until it has  $\mu_{i-}$  length, and we rename the indexes of the springs such that they are sorted in ascending order with respect to  $f_i$ .

For a contraction smaller than  $l_0$  the chain reacts as a linear spring with constant:

$$K_0 = \frac{1}{\sum_{i=0}^u \frac{1}{k_i}}$$

At this point there is a discontinuity due to the fact that the spring  $s_0$  cannot be further contracted, so the stiffness for this new configuration is:

$$K_1 = \frac{1}{\sum_{j=1}^h \frac{1}{k_j}}$$

and the length of the next spring that will be reduced to its minimal length is:

$$l'_1 = l_1 - \frac{f_0}{k_1}$$

Generalizing we obtain:

$$K_i = \frac{1}{\sum_{g=i}^h \frac{1}{k_g}}$$

$$l'_i = l_i - \frac{f_{i-1}}{K_i}$$

The relation between spring contraction and the force applied is shown in Figure 4.b. We define the stiffness constant in contraction as the slope  $\bar{K}$  of the straight line which subtends the same area of the one subtended by the piecewise linear function. In other words, we replace the chain of springs with a single spring which stores the same energy when contracted to a length equal to the sum of all the minimum lengths of the spring chain and having the same length of the chain at its rest position. Hence the formula for  $\bar{K}$  is:

$$\bar{K} = \frac{f_u}{l'_u}$$

(details on the formula above are in [11]).

In the case of the spring elongation, we can do the same considerations obtaining the same kind of relation: if we define  $\mu_{i-} = l_i - \delta$  and  $\mu_{i+} = l_i + \delta$ , we obtain *exactly* the same relation.

If we intend to model fracture, i.e. when a spring breaks up when trespassing its maximum elongation, the constant of stiffness of the chain is simply  $K_0$  until the

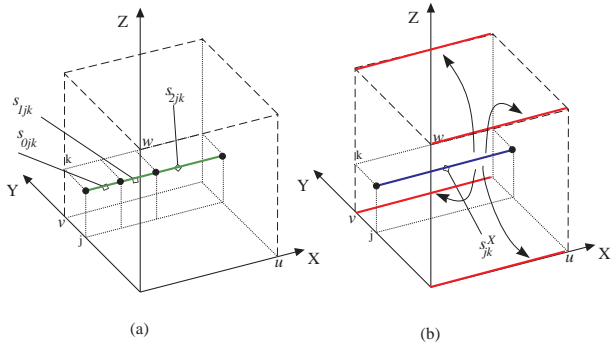


Figure 5: Computation of the characteristic parameters of the four X-aligned spring as a composition of the set of X-aligned spring.

chain reaches its maximum elongation (the minimum between the maximum elongation of all of the springs in the chain). As in the case of contraction, we define the stiffness constant in elongation as the slope of the straight line that best approximates the piecewise linear function obtained in this case.

Finally, we define the constant of stiffness  $k^X$  of the single spring representing the chain as the mean value of the constant in contraction and elongation.

### Maximum elongation

If we do not want to model fracture, the maximum length of a chain of spring is the sum of the maximum length of all the springs; otherwise, it is the length of the chain when the the first spring breaks up. Considering the springs in ascending order with respect to  $k_i(\mu_i^+ - l_i)$ :

$$\mu^{X+} = \frac{k_0(\mu_0^+ - l_0)}{K_0}$$

### Merging X-aligned springs

By simplifying series of X axis-aligned chains of springs in single springs we obtain a set of X axis-aligned parallel springs (see Figure 5.a). We must now determine the value of the four X-aligned springs which defines the behaviour of the simplified cell  $\gamma$ . These four springs are obtained by distributing the stiffness values of each composed X-aligned spring between them, proportionally to the distance (and analogously to the approach adopted for the determination of the masses of  $\gamma$ ):

$$K_{s_{\bar{x}, y, z}} = \sum_{j,k=0}^{vw} ((1-y) \cdot \frac{j}{u} + y \cdot \frac{u-j}{u}) + ((1-z) \cdot \frac{k}{w} + z \cdot \frac{w-k}{w}) K_{s_{jk}^X}$$

In the same way, the maximum and minimum lengths are in turn distributed, using the same weight coefficients.

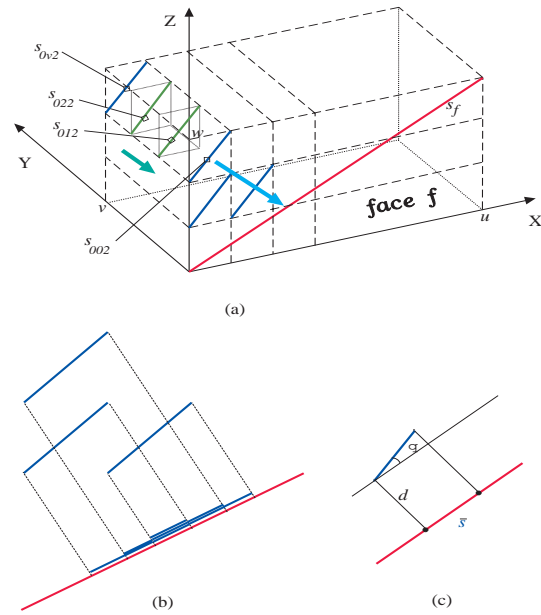


Figure 6: All the internal face shear springs are added on the corresponding shear spring on face  $f$ , on the  $XZ$  plane (a); then, all these composed shear springs on face  $f$  are projected (c) and composed (b) on the face shear spring  $s_f$ .

### 4.3.3 Shear springs of $\gamma$

The determination of the defining constants ( $k, \mu_-, \mu_+$ ) for the shear springs that characterize the simplified cell is quite more complicated than for the axis aligned ones. In the following, we describe only how we determine the **face shear** springs for the two faces lying on the plane  $YZ$ , i.e. spring  $s_f$  in Figure 6 which joins the vertex at the origin and the vertex at the point  $(u, 0, w)$ .

We define the constants for this spring as a function of the constants of all the springs equally oriented in  $\Gamma$ , again by adopting a two-phases process:

1. each pair of  $X = i, Z = k$  coordinates determines an associated set of cells (e.g. the stick of cells which are adjacent in the  $X$  direction). We distribute the value of each spring  $s_{ijk}, j = [0, \dots, v]$  internal to this stick to the two equally aligned springs lying on the two external faces of the cell stick ( $s_{i0k}$  and  $s_{ivk}$ , see Figure 6.a), depending on their relative distances from the external faces (we use a formula similar to the one used to redistribute the masses);
2. then, we define the shear spring  $s_f$  relative to the simplified cell as a function of all the ‘‘composed’’ shear springs  $s_{i0k}$  computed for each cell stick.

The value of each generic spring  $s_{ijk}$  is projected on the straight line parallel to  $s_f$  and passing for an extreme of  $s_{ijk}$  (see Figure 6.b). The projected endpoints of the composed shear springs divide shear spring  $s_f$  in intervals; we assign the contribution of

each spring  $s_{ijk}$  to all the intervals onto which it projects, proportionally to the length of the intervals and to the distance  $d$  from spring  $s_f$  (see Figure 6.c). In conclusion, the contribution of a “composed” shear springs  $s_{i0k}$  to the stiffness  $\bar{k}$  of a given interval  $\bar{s}$  on shear spring  $s_f$  is:

$$k_{\bar{s}} = k_{s_{i0k}} \cdot \cos \theta \cdot \alpha d$$

where  $\alpha$  is a weighting factor for distance  $d$ .

Once all “composed” shear springs have been projected on  $s_f$ , we obtain a partition of  $s_f$  in a set of intervals (see Figure 6.b), such that we can consider now  $s_f$  as the composition a chain of springs. The characteristic parameters (min, max, rest length and stiffness) of spring  $s_f$  can now be computed following the same approach proposed in Subsection 4.3.2.

The step 2 is adopted also to compute the **body shear springs** of the simplified cell.

Since the springs and the masses on the border of  $\Gamma$  that *do not* belong to the border of the whole object are shared with their adjacent cells, in the simplification step we consider the values of the masses and of the constant of stiffness of such springs as half of their true value.

The determination of a bound of the error that can be introduced in the computation of the characteristic of the simplified fragments is not easy, in particular for the shear springs. An empirical verification is therefore mandatory, and some details on the on-going experimentation are presented in the last section.

#### 4.4 Choosing a correct representation

Once the physic behaviour of the mass-spring cells corresponding to the octree nodes have been determined, we have to define some rules for extracting different resolution representations from this multiresolution model. Analogously to the more general MT scheme, the set of cuts on the octree is isomorphic to the set of all the different representations obtainable combining the cells corresponding to the nodes of the octree such that the domain is completely covered and there is no proper intersection between the cells. Hence choosing a cut means to choose a representation.

Two distinct goals can be considered:

##### 1. Static representation

In this case the representation is chosen and fixed for all the duration of the simulation; it can depend on:

- *the relative importance of objects or sub-components*: in a surgery simulation scene, for

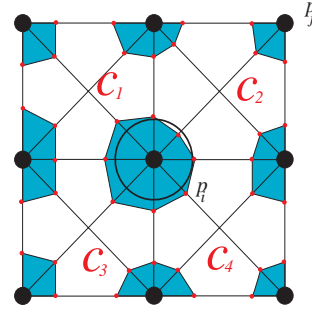


Figure 7: The circle involves points of  $c_2$  that are not in the safe box, hence  $c_2$  is marked as unsafe

example, greater detail can be given to the organ on which the surgeon is going to operate, and less to organs/tissues far away from the virtual surgery focus;

- *the available computational power*; more powerful is the system available, finer can be the model used.
- *the magnitude of the applied forces*. As we stated in Section 2, the same unconstrained mass spring system cannot be used for every magnitude of force tensors without experimenting what we called *cell reversion*. Assuming to know the maximum magnitude  $\|f\|_{max}$  of the forces that can be applied, in order to choose a correct representation to react to such force we proceed as follows.

In a first step, we define the *safe box* for each mass point  $p_i$ , that is the polyhedron space surrounding  $p_i$  where it can move in a single time step without causing any cell reversion. The vertices of such box (see Figure 7) lie along each spring  $s_{ij}$  incident on  $p_i$  at a distance of  $\beta(i, j) \cdot \|p_i - p_j\|$  from  $p_i$ ; note that  $\beta(i, j) + \beta(j, i) < 1$  guarantees that no reversion happens. After this step, for each mass point we apply the following algorithm:

- calculate the displacement of a point in a single time step as  $\delta = \|f\|_{max} \cdot \frac{\Delta t^2}{2m_p}$
- for each cell  $c$  having  $p_i$  as one of its vertices: if the “octant” of the sphere centered at  $p_i$  with radius  $\delta$  lying in  $c$  is not entirely contained in the safe box, mark  $c$  as *unsafe* (Figure 7 shows a 2D example where cell  $c_2$  is unsafe)

At the end we will obtain a set of *unsafe* cells. What we do is to choose a cut on the octree such that each unsafe cell is replaced by a bigger one and to repeat the algorithm until no cells result unsafe.

##### 2. Dynamic representation

A simple manner to obtain a dynamic change of res-

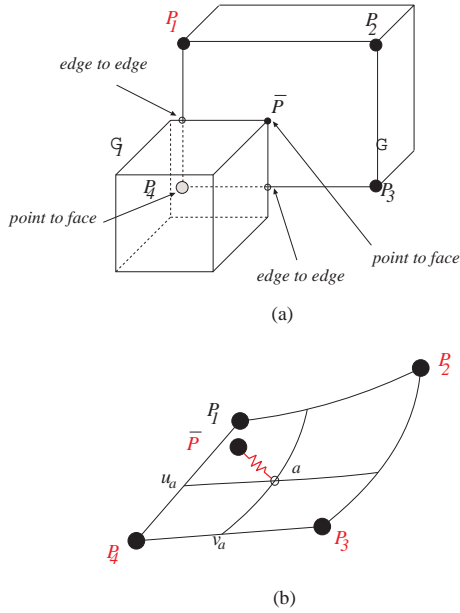


Figure 8: Adjacent cells in a variable resolution mass-spring system might present a more complex component relations (a); spring added to model *point face* relation are shown in (b).

olution depending on the applied forces can be to run the simulation starting with the maximum resolution (or a statically obtained variable resolution), and by progressively update the cut by replacing all the cells which became unsafe with higher resolution cells. Several difficulties arise in trying to obtain visual and physical smoothness while resolution is dynamically updated, and we consider therefore premature to formulate a detailed solution here.

## 4.5 Error evaluation

Representations with different levels of detail behave in different manner. We plan to run a simple experiment to compare two different representations  $\Gamma_a$  and  $\Gamma_b$  when subjected to the same input forces. We will introduce a set of uniformly distributed landmarks  $G = \{g_i, i = 0 \dots, n\}$  on the surface of both the representations when they are in their rest (equal) shape, and measure the difference of behaviour at the time  $t$  as the sum of the modules of distance between corresponding landmarks:

$$Err(t) = \sum_{i=0}^n \|g_i^a - g_i^b\|.$$

The position of the landmarks does not have to coincide with the position of the mass points, but can lie in any point of a face, and the coordinates of each landmark are expressed as a function of coordinates of the vertices that define the face where the landmark lies.

## 4.6 Managing continuity in a variable-resolution mass spring system

Since not all of the parts of the object will be in general represented with the same resolution, adjacent cells will be different in size, and hence some mass points might now not present a corresponding incident spring with respect to the adjacent cell (see Figure 8.a). A mass point of a given cell may correspond with either: (a) a mass point of the adjacent cell, (b) a point on a face of the adjacent cell or (c) a point on a spring of the adjacent cell. It appears evident that this system is not a mass spring system according to the original definition presented in Section 2. Therefore we have to extend the definition of the mass spring system with a new *point-face* relation (and the *point-edge* is considered as a degenerate case of the latter). Consider the cells  $\Gamma_1$  and  $\Gamma_2$  depicted in Figure 8. The mass point  $\bar{P}$  is obviously connected with the other three mass points of cell  $\Gamma_1$ , but it has no direct connection with the mass points of the cell  $\Gamma_2$ . If we run a simulation for this system then we do not want the two cells to be free to deform and to move independently, because they represent the same solid. Consequently, point  $\bar{P}$  has to be constrained to lie on the face  $f$  of  $\Gamma_2$  defined by  $P_1, P_2, P_3$  and  $P_4$ . In order to do that, we add a spring between the mass point  $P_1$  and the point  $a$  on the face, corresponding to position of  $P_1$  at the rest configuration. Clearly the coordinates for  $a$  vary with the ones of the points delimiting the face. The simpler surface interpolating four points in the space is a bilinear patch, hence we define the face  $f$  as:

$$f = \{p_1 \cdot (1-u)(1-v) + p_2 \cdot u(1-v) + p_3 \cdot uv + p_4 \cdot (1-u)v \mid u, v \in [0, 1]\}$$

and the point  $a$  corresponds to  $(u_a, v_a)$  in the parametric space  $[0, 1]$ , that are calculated at the rest shape. The spring added, when contracted or elongated, exerts a force on the mass point  $\bar{P}_1$  and on the face  $f$ , that we have to distribute on the points  $P_1, P_2, P_3$  and  $P_4$ . These forces are calculated considering the face as a rigid body without mass and resolving a simple static equilibrium problem: given a force  $F$  acting on the point  $a$ , find the forces  $F_1, F_2, F_3$  and  $F_4$  balancing  $F$  (details in [11]).

The *edge to edge* and the *point to edge* relations are managed with the same approach. In this case the common point is given by a simple linear interpolation. Note that if two cells share a face in the rest position, the springs added to manage *point to face*, *edge to edge* and *point to edge* relations tend to maintain the faces of cells stitched together.

## 5 Conclusions and future work

The main contributions of the paper is the introduction of a multiresolution approach in context of Deformable Object

Modeling. In particular, we have investigated techniques for the construction of a multiresolution representation of a mass spring system, based on a simple octree-based decomposition. This involves the definition of a methodology for deriving the characteristic parameters of a simplified cell from those of the replaced cells (given a simple replacement rule based on axis-aligned hexahedra), and the extension of the mass spring model with ad hoc introduced *point to face* and *point to edge* relations. Furthermore, we give a criteria to extract a representation depending on the range of force intensities which can be applied to the object modeled, such that each part of the object is represented at the appropriate resolution.

A prototypal implementation of the approach proposed is currently in advanced development state.

In order to extend the applicability of the proposed approach to a wider and more general context, further research is needed to support not only hexahedral decompositions (which are a obvious choice in the case of voxelized domain) but also irregular simplicial decomposition. Moreover, modeling also other more complex physical behaviours, for example plasticity and viscosity, could be taken into account. These extensions require a more general technique for deriving physical characteristics of simplified fragments, able to manage generalized coarser representations of a mass spring systems.

## References

- [1] D. E. Breen, D. H. House, and P. H. Getto, *A physically-based particle model of woven cloth*, The Visual Computer **8** (1992), no. 5–6, 264–277.
- [2] D. E. Breen, D. H. House, and M. J. Wozny, *Predicting the drape of woven cloth using interacting particles*, Computer Graphics **28** (1994), no. Annual Conference Series.
- [3] Morten Bro-Nielsen and Stephane Cotin, *Real-time volumetric deformable models for surgery simulation using finite elements and condensation*, Computer Graphics Forum **15** (1996), no. 3, C57–C66, C461.
- [4] Marie-Paule Cani-Gascuel, *Layered deformable models with implicit surfaces*, Graphics Interface, June 1998.
- [5] H.-G. Krumm Ch. Kuhn, U. Khnapfel, *A 'virtual reality' based training system for minimally invasive surgery*, C.A.R. Proceedings (Paris, France), June 1996.
- [6] David T. Chen and David Zeltzer, *Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method*, Computer Graphics **26** (1992), no. 2, 89–98.
- [7] Sabine Coquillart, *Extended free-form deformation: A sculpturing tool for 3D geometric modeling*, Computer Graphics (SIGGRAPH '90 Proceedings) (Forest Baskett, ed.), vol. 24, August 1990, pp. 187–196.
- [8] L. De Floriani, P. Magillo, and E. Puppo, *Efficient implementation of multi-triangulations*, Proceedings IEEE Visualization 98 (Research Triangle Park, NC (USA)
- [9] L. De Floriani, E. Puppo, and P. Magillo, *A formal approach to multiresolution modeling*, Geometric Modeling: Theory and Practice (R. Klein, W. Straßer, and R. Rau, eds.), Springer-Verlag, 1997, pp. 302–323.
- [10] David R. Forsey and Richard H. Bartels, *Hierarchical B-spline refinement*, vol. 22, August 1988, pp. 205–212.
- [11] Fabio Ganovelli, *Models for deformable object modeling*, Tech. Report TR-??, Istituto per L'Elaborazione dell'informazione - National Research Council, January 31.
- [12] Sarah F. Gibson, *3D chainmail: a fast algorithm for deforming volumetric objects*, Proceedings of the Symposium on Interactive 3D Graphics (New York), ACM Press, April 27–30 1997, pp. 149–154.
- [13] E. Keeve, S. Girod, and B. Girod, *Computer-aided craniofacial surgery*, Journal of the Intern. Society for Computer Aided Surgery **3** (1996), no. 2, 6–10.
- [14] D. Meagher, *Geometric modeling using octree encoding*, Computer Graphics and Image Processing **19** (1982), no. 2.
- [15] S. M. Platt and N. I. Badler, *Animating facial expressions*, ACM Computer Graphics **15** (1981), no. 3, 245–252.
- [16] Xavier Provot, *Deformation constraints in a mass-spring model to describe rigid cloth behavior*, Graphics Interface '95, May 1995, ISBN 0-9695338-4-5, pp. 147–154.
- [17] E. Puppo and R. Scopigno, *Simplification, LOD, and Multiresolution - Principles and Applications*, EUROGRAPHICS'97 Tutorial Notes (ISSN 1017-4656), Eurographics Association, Aire-la-Ville (CH), 1997 (PS97 TN4).
- [18] S. H. Martin Roth, Markus Hans Gross, Silvio Turello, and Friedrich Robert Carls, *A Bernstein-Bézier based approach to soft tissue simulation*, Computer Graphics Forum **17** (1998), no. 3, 285–302.
- [19] T. W. Sederberg and S. R. Parry, *Free-form deformation of solid geometric models*, Computer Graphics **20** (1986), no. 4, 151–160.
- [20] John M. Snyder and Adam R. Woodbury, *Interval methods for multi-point collision between time-dependent curved surfaces*, SIGGRAPH, vol. 18, ACM Press, 1993.
- [21] D Terzopoulos, J. Platt, A. Barr, and K. Fleisher, *Elastically deformable models*, Computer Graphics, June 1987 **21** (1987), no. 4, 205–214.
- [22] D. Terzopoulos and K. Waters, *Physically-based facial modelling, analysis, and animation*, The Journal of Visualization and Computer Animation **1** (1990), no. 2, 73–80.
- [23] Demetri Terzopoulos and Kurt Fleischer, *Modeling inelastic deformation: Viscoelasticity, plasticity, fracture*, Computer Graphics (SIGGRAPH '88 Proceedings) (John Dill, ed.), vol. 22, August 1988, pp. 269–278.
- [24] P.Faloutsos M vande Panne D.Terzopoulos, *Dynamic free-form deformations for animation synthesis*, IEEE Transactions on Visualization and Computer Graphics **3** (1987).
- [25] R.M. Koch M.H. Gross F.R.Carls D.F. von Buren G. Fankhauser and Y.I.H. Parish, *Simulating facial surgery using finite element models*, Computer Graphics **30** (1996), no. Annual Conference Series, 421–428.
- [26] K. Waters, *A muscle model for animating three-dimensional facial expression*, Computer Graphics (SIGGRAPH), Volume 21, Number 4, July 1987, pp. 17–24.