Volume 19 (*2000*), *Number 3*

A Multiresolution Model for Soft Objects supporting interactive cuts and lacerations

Fabio Ganovelli[†] , Paolo Cignoni[‡], Claudio Montani[§] and Roberto Scopigno[¶]

Istituto Elaborazione dell'Informazione[‖], C.N.R., Pisa, Italy

Abstract

Performing a really interactive and physically-based simulation of complex soft objects is still an open problem in computer animation/simulation. Given the application domain of virtual surgery training, a complete model should be quite realistic, interactive and should enable the user to modify the topology of the objects. Recent papers propose the adoption of multiresolution techniques to optimize time performance by representing at high resolution only the object parts considered more important or critical. The speed up obtainable at simulation time are counterbalanced by the need of a preprocessing phase strongly dependent on the topology of the object, with the drawback that performing dynamic topology modification becomes a prohibitive issue.

In this paper we present an approach that couples multiresolution and topological modifications, based on the adoption of a particle systems approach to the physical simulation. Our approach is based on a tetrahedral decomposition of the space, chosen both for its suitability to support a particle system and for the ready availability of many techniques recently proposed for the simplification and multiresolution management of 3D simplicial decompositions. The multiresolution simulation system is designed to ensure the required speedup and to support dynamic changes of the topology, e.g. due to cuts or lacerations of the represented tissue.

1. Introduction

Two main factors stimulate the quest for improved efficiency of medical surgical simulation: the need to disseminate virtual tools for rehearsal and training to a wider community, and the need to support an ever increasing realism. The first point is made easier by the impressing performance improvement of personal computers, in terms of both pure numerical speed and graphics throughput. Graphics PCs can bring virtual simulation to the global community of medical schools and hospitals, to support both surgery didactic and surgery planning. The need of this type of tools is made more pressing by the diffusion of laparoscopic surgery, that requires surgeons skilled in controlling the status of the operation and the location of the surgery instrumentation by decoding the information coming from an electronic output device, where video images replaces the direct inspection of the surgery field.

The second point, i.e. the high realism required, forces to adopt complex digital models. The more fidelity we need in the simulation of the anatomical organs, the more complex is the representation required and the higher are the processing times. But realism is not only a matter of data resolution. Analogously to other virtual simulators, virtual surgery systems requires a very high update frequency, e.g. at least 10-15 Hz for visual feedback and much more for force feedback. The efficiency of the simulator is therefore highly critical, especially if the target computing unit is a PC.

Therefore, we have a number of contrasting constrains: high fidelity, which in general implies accurate models and high frame rate, and use of low cost computers. A number of proposal have been recently presented to fulfill these objectives. But even if the efficiency of the simulation models have been largely improved in the last few years, soft object deformation remains a rather complex task and can be solved in interactive time only on models composed by a few hundreds of

[†] Email: ganovell@iei.pi.cnr.it
[‡] Email: cignoni@iei.pi.cnr.it
[§] Email: montani@iei.pi.cnr.it
[¶] Email: roberto.scopigno@cnuce.cnr.it
[‖] Loc. S. Cataldo, 56100 Pisa, ITALY

cells. Moreover, the more are the effects that have to be simulated (deformation, cuts, lacerations, etc), the more complex is in general the simulation model required. To improve efficiency, an hybrid model was recently proposed, which applies different models to different areas of the objects manipulated: because the deformation model that supports tissue cuts is more time-expensive, it is then applied only to the area where cut actions have been planned in a pre-operating phase [4].

We proposed a different approach. Given a deformation model, we adapt the resolution of the data description to reach the goal of speeding up the simulation process and [hopefully] producing results very near to the one obtained on the full resolution model. The idea is to adopt a multiresolution approach, and to support dynamic construction of optimized level of detail (LOD). In the case of a surgical simulation system, the resolution of a given LOD has to be proportional in each instant of time to the proximity of the surgical focus area, allowing a very precise representation in the areas directly affected by the surgery action, and degrading data resolution smoothly with the increase of the distance from the focus.

Dynamic LOD have been intensely studied, but mainly to support interactive visualization (where the driving factor is the viewer position). Here the focus is on user interaction: the interesting area is the one the user is manipulating (or is going to manipulate). Therefore, the selection of the optimal LOD has to be driven by a more complex set of parameters and heuristics.

It is well known that the management of dynamic LOD introduces an overhead, both in time and space. For this reason static LOD are often preferred in interactive visualization system (e.g. navigation in urban sites). Fortunately, a user of a surgery simulation system drives smoothly the modification in space of the focus region, at least much more than users of driving simulators or computer games. The location of the surgical instruments is updated smoothly, both to move with caution in the operating area and to perform controlled cut/manipulation actions. This give us the possibility to avoid an expensive 1:1 ratio between frame rate and update rate of the dynamic LOD model. The current LOD is in general adequate for a number of time steps, and therefore the time overhead required for its creation from the multiresolution model can be amortized on a longer life cycle. The efficiency of the multiresolution approach adopted (dynamic LOD extraction cost is in the order of 50 milliseconds, see the results section) makes the overhead affordable.

We propose therefore the adoption of a multiresolution approach to speedup soft tissue simulation, specifically oriented to surgery simulation needs. The paper is organized as follows. A brief overview of the previous works is in Section 2. The physical model adopted, based on tetrahedral decomposition, is described in Section 3. The central part of the paper starts from Section 4 where we briefly introduce the MT multiresolution framework and describe in more detail how do we evaluate the approximation error and how do we extract dynamically the LOD representations. Section 5 presents the approach devised to perform the topological modification induced on the MT structure by cuts or lacerations. The following two sections, 6 and 7, describe the cut and laceration issues. Results of a prototypal implementation are reported in Section 8 and conluding remarks are drawn in Section 9.

2. Previous Works

Several approaches have been proposed to model realistic interaction with deformable objects in these last years. The classic compromise is between accuracy and real-time performance. Accurate models, for example, are used for surgery planning: models for facial surgery were proposed in [11, 16], where the skin is modeled with a triangular surface connected with the bone by introducing one-dimensional springs. Here the use of the Finite Element Method [19] gives physically consistent results, but requires an amount of computational time not compatible with interactive simulation. Other approaches adopting FEM [5, 2] support a nearly interactive frame rate but using a small number of elements. However, the use of FEM require a preprocessing phase strongly depending on the topology of the object, hence preventing the possibility to cut the object.

Less accurate but faster and simpler models are based on particles system [15]: here the volume is modeled as a set of mass points and a set of relations between mass points (this model is often referred as Mass Spring System). Particle systems present two main advantages: they are generally very easy to implement and they do not require initialization. For these reasons they are widely employed in real-time surgery simulation. On the other hand, the computation required depends on the number of particles used. For this reason, a current trend in this field is to introduce multiresolution representation to use selectively more particles where an higher accuracy is required, e.g. in proximity of the action focus [8, 7]. The capability to perform cuts is a fundamental feature of a virtual surgery simulator, and is also an example of focused action. Delingette and others have recently presented a new approach that supports efficiently the dynamic modifications of anatomical structures [4]. Assumed that dynamic models are more expensive in time than optimized static approaches, they propose to decompose the anatomical structure in different sections: the one onto which we plan to perform cuts or modification during the simulated surgical procedure, and the rest of the anatomy that will not be affected by those cuts. Therefore, they propose to apply different elastic models to the different sections of the anatomy: the more expensive dynamic model is then applied only to the section that the user will probably cut. This approach presents some drawbacks. Ensuring continuity and smoothness on the interconnecting boundary between two sections can be not straightforward. Moreover, we have to plan in advance and very precisely which subsection of the anatomy will be affected by the dy-

namic modification actions. This planning might not be easy, also because we have to define precisely the boundary of a 3D subvolume. Efficiency gains are directly dependent on how much this selection is tight with respect to the area actually modified during the simulation. Moreover, cuts and lacerations are implemented on tetrahedral decompositions by simply removing the tetrahedra cells involved in the cut. This is a simple and straightforward approach to mesh update, but it requires the adoption of a very high resolution representation to produce a sufficiently realistic effect.
The accuracy of the cut can be improved by splitting every tetrahedron involved in the cut into a set of smaller cells [1]. The only drawback of the approach proposed in [1] is that the fixed splitting scheme adopted produces an excessive number of small tetrahedra in the proximity of the cut.

The dynamic modification of the mesh (e.g. due to cuts or lacerations) is not supported by most of the multiresolution approaches. This is because multiresolution is often implemented by static data structures (for example octrees, where each node defines a representation coarser than those of its children). Hence, when a cut is performed it generally affects not only the current mesh but also the multiresolution data structure. By our knowledge, none of the proposed multiresolution frameworks supports dynamic updates when the topology of the object changes, except the one proposed here.

3. Physical Model: a Particle System Defined on a Tetrahedral Decomposition

A particle system is composed by a set of elements (the particles) and a set of relationship between the elements. Each particle is basically described by its position, mass and velocity. A set of relations defines the interaction between the particles involved. The behaviour of the system is governed by the well known Newtonian second law:

$$M\ddot{x} + D\dot{x} + Kx = f \tag{1}$$

Our system relies on the availability of:

- a three dimensional dataset defining a density function $\sigma : \mathbb{R}^3 \rightarrow \mathbb{R}$
- a tetrahedral decomposition Γ of the subset of $\mathbb{R}^3$ space that represents the object of interest.

Given a tetrahedral mesh, the associated particle system is simply obtained by considering each mesh vertex as a particle and each edge as a linear elastic relation between the two particles at its extremes. The first preprocessing step, that we describe in the next paragraph, sets the physical parameters of the system.

3.1. Definition of the mass values

To assign the mass values to the particles we require that:

- for each tetrahedron $\tau \in \Gamma$ with vertices/particles (p_0, p_1, p_2, p_3) located in $(x_0, x_1, x_2, x_3) \in \mathbb{R}^3$ and having masses (m_0, m_1, m_2, m_3):

$$\int_{V(\tau)} \sigma(x)dx = \sum_{i=0}^{3} m_i$$

  where $V(\tau)$ is the volume of cell τ;
- the total mass of the object should remain the same even if different space partitions are adopted.

Each tetrahedron contributes to the mass values of its vertices. Since a vertex p is shared by n tetrahedra, with n the number of tetrahedra in the representation incident in n, its mass value will be given by the sum of all n tetrahedra contributions. We define the contribution of cell τ to the mass m_i of each one of its vertices as:

$$\int_{V(\tau)} \sigma(x)c_i(x)dx$$

where c_i are the coefficient of the convex combination of the point x_i which gives point x.

It is trivial to see that this assignment respects the conditions imposed.

3.2. Definition of the stiffness parameters

A methodology for the definition of the stiffness value of the springs of a triangle cell, in the case of a membrane modeled with a triangle mesh, has been proposed in [18]. It can be simply generalized to cope with the springs of the tetrahedral cells used to represent a volume. If e is an edge/spring of a tetrahedron τ, $stif_{e,\tau}$ is the contribution of the tetrahedron τ to the stiffness $stif_e$ of edge e:

$$stif_{e,\tau} = \frac{E\ V(\tau)}{|e|^2}$$

where E is the *Young's modulus* of the material, supposed to be constant throughout the tetrahedron. Therefore, the stiffness of edge e is obtained by summing all of the contributions of the tetrahedra incident on e:

$$stif_e = \sum_{\tau \in Incid(e,\Gamma)} stif_{e,\tau}$$

When the parameters are set, the system can be integrated with a classic PDE solver, e.g. with a fourth order Runge-Kutta method.

4. Use of MT for Multiresolution Representation

The Multiresolution Triangulation (MT) is a general and dimension-independent framework introduced in [6, 14]. The general idea underlying the MT is that any multiresolution mesh can be built through local operations that progressively modify an initial mesh through either refinement or simplification, and that local modifications can be arranged into a

partial order according to their dependencies. For example, a simplification software permits to reduce progressively an input mesh size by performing a series of atomic incremental update steps; every incremental update in general reduces mesh size and increases the approximation of the current representation with respect to the initial mesh. A mesh update consists of the replacement of a set of cells, called *fragment*, with a new fragment, denoted with Σ_i, composed by a smaller number of elementary components. By definition, the replaced and the new fragment share the border. The simplification process induces a partial order among fragments, which can be represented with a Directed Acyclic Graph (DAG). The main idea of the MT is that we may represent the whole simplification process by storing all the fragments produced (either explicitly or implicitly) and all the inference relations between fragments (relative to their partial order).

The MT DAG has exactly one node with no entering arcs, the *root* fragment Σ_r, that corresponds to the mesh at the end of simplification process, and a set of nodes with no leaving arcs, that corresponds to fragments formed only by tetrahedra of the input mesh. A *cut* is a collection of arcs such that each path from the root to a leaf has exactly one arc belonging to the cut, and a valid representation corresponds to each cut. Given an MT representation, we can extract a given representation of the mesh by simply defining the associated cut in the MT DAG.

Given an MT, we can extract a representation (a mesh) with a continuum varying LOD in only a few milliseconds, also in the case of large meshes. This is done by specifying a *query*, that is a function from $\mathbb{R}^3$ to $\mathbb{R}$ that specifies, for each point, what is the maximum *approximation error* that should be verified by the extracted LOD. The approximation error is a real value associated to each tetrahedron, that estimates the representation accuracy with respect to the associated portion of the original mesh.

4.1. The approximation error

The multiresolution model is built in a pre-processing phase, using known approaches based on 3D simplicial meshes decimation [3, 17]. The evaluation of the approximation error is a crucial point in any multiresolution model. In our case, due to the particular application field, we prefer to assign to each cell in the MT an error evaluation rule that takes into account the size of the cells. We do not describe here in detail the simplification and multiresolution construction phase, for the sake of conciseness. Let us only say that simplification is based on iterative updates and is driven by a peculiar error evaluation heuristic:

- vertices on the boundary of the 3D mesh are simplified by using an error metric based on the approximated Hausdorf distance from the boundary of the original input mesh, weighted by the size of the incident simplicial cells;
- vertices in the interior of the 3D mesh are simplified by using an error metric which considers only the size of the incident simplicial cells (i.e. smaller cells are removed first).

Since a decimation step consists of replacing a set of cells with a smaller set which span the same subvolume, the *average* volume of the single cells created during a sequence of decimation steps strictly increase. We set the error of each fragment Σ_i in the MT equal to the average volume of its tetrahedra:

$$\varepsilon(\Sigma_i) = \sum_{\tau \in \Sigma_i} V(t) \; \frac{1}{\#\Sigma_i}$$

where $V(\tau)$ indicates the volume of a cell τ.

4.2. Extracting a dynamic LOD representation

Our multiresolution approach has been designed to represent environments composed by several objects. An example is a simulator for abdominal surgery, with multiple organs represented. In this context, we want to select dynamically the resolution, assigning higher priority to the parts of the object which are in the current action focus. In our vision, such focus is the zone in the proximity of the surgical tool. Therefore, the system should support the dynamic selection of a level of detail (LOD) such that resolution is maximum near to the focus region and decreases as we go away from it. Since the focus changes during the simulation, the LOD should reflect this evolution more timely as possible. The state of the system at time t, that we denote with S_t, is the collection of the velocity and the acceleration of all the particles belonging to the LOD representation at that time. The way to infer the state $S_{t+\delta t}$ from S_t, when the representation is the same, is to integrate the equations of motion over the step δt.

The situation changes when at time $t+\delta t$ the representation Γ_t is updated by extracting a new LOD representation Γ_{t+1} from the multiresolution model, because:

- new particles that do not belong to the previous representation Γ_t are introduced;
- some particles that belong to the previous representation can be not included in the current one.

The problem is how to infer the velocity and the acceleration of the particles introduced at time $t+\delta t$. This is a fundamental point in the implementation of a multiresolution soft tissue system [7, 13, 8] and the typical solution is to interpolate the unknown values (i.e. the state) from particles present in the representation at the previous step. Let p_i be one of the particles introduced at time $t+\delta t$. To interpolate its state we should detect the corresponding cell τ in Γ_t onto which the state of p_i should be interpolated. Our MT is built by decimation in a preprocessing phase; at each decimation step a vertex p (i.e. a particle) is removed and the polyhedron given by the union of the tetrahedra incident in p is retriangulated. Given the unknown particle p_i, the MT construction rule implies that one of the tetrahedra in the fragment originated by

the removal of p_i, say τ', should contain the vertex/particle p_i. If τ' is contained in the representation Γ_t, then we can interpolate the state of the particle p at state t by using the state at the vertices of τ'. Otherwise, we continue recursively the visit of the MT structure, searching in the following decimated fragment a cell τ'' such that it contains p and is part of Γ_t.

5. Applying Topological Modifications

As we introduced earlier, the capability of updating a representation depending on user-driven topological modifications is a fundamental feature of a surgical simulation system. In this context, the user drags a scalpel in the virtual scene, that is modeled with a segment. The movement of the scalpel's blade defines a *swept surface*. When the swept surface intersects the object, it intersects a subset Γ_{cut} of its tetrahedra. The modification required on the MT structure by this event consists of:

- replacing each tetrahedron in Γ_{cut} with a set of tetrahedra having no proper intersection with the swept surface;
- updating the MT structure.

The first update, treated in Section 5.2 is simpler. Here multiresolution issues are not involved and we have only to find an efficient solution for cell clipping, and then to assign masses and stiffness values to the particles and edges of the new tetrahedra introduced. The second update is briefly summarized in the next section and extensively described in [9].

5.1. Updating the MT

As we stated in Section 4, a fragment Σ_i of the MT is a set of tetrahedra. Furthermore, since such set corresponds to the tetrahedra added in a decimation step, it is a simply connected simplicial complex. The basic idea for the MT update (supposing in the description that the fragment is split in two sections by the cut) is as follows. Given the surface plane S swept by the scalpel blade, that divides in two parts the volume $V(\Sigma_i)$ covered by the fragment Σ_i, we first divides the cells in Σ_i in three sets: the cells on the left of plane S are stored in ${\Sigma_i}^1$; those on the right are stored in ${\Sigma_i}^2$; those intersected by S are split along S, and the new cells resulting are returned either in ${\Sigma_i}^1$ or ${\Sigma_i}^2$. At the end of the process, fragment Σ_i is replaced by the two fragments ${\Sigma_i}^1$ and ${\Sigma_i}^2$ such that: $V({\Sigma_i}^2) \cup V({\Sigma_i}^2) = V(\Sigma_i)$ and all cells on ${\Sigma_i}^1$ are on the same side of S (and viceversa for ${\Sigma_i}^2$).
As a consequence of this operation, all the relations of the MT that involve the fragment Σ_i have to be updated.
We omit here the technical details (see the paper cited above) and observe that the update of the MT requires a number of operations linearly proportional to the number of fragments that overlap the volume intersected by the sweeping surface. In other words, it does not depend directly on the total size of the MT but on the size of the section intersected by the sweeping surface. Furthermore, if we have strict time constrains, we can temporarily postpone the update of the MT DAG. Tissue simulation requires only a prompt modification of the current LOD topology (that is, all fragments contained in the current LOD which are intersected by the current cut have to be split before the simulation of the next deformation step can start). On the other hand, the MT DAG update can be performed in background, and we only have to ensure that the next LOD extraction will not be performed before the DAG update has been completed.

5.2. Decomposition of tetrahedra

A solution for cell cutting was proposed recently [1]. To support the interactive execution of the splitting phase, for any cut occurrence the authors adopt the same standard 1:17 splitting scheme (computed in a preprocessing phase) for creating the new tetrahedra. Though such splitting pattern covers all the possible configurations arising from any partial or complete cut, the number of tetrahedral cells generated is never the optimal one, and the mesh is highly fragmented after only a few cuts. They also distinguished between the different types of cut, using a Look Up Table to know which faces have to be introduced to represent the new surface created by the cut.
We have recently proposed [9] a more efficient solution. We used a look up table also for the cell splitting that gives, for each type of cut, the optimal tetrahedra decomposition associated to the cut. The table is indexed by the pattern of intersections between the current cell edges and the splitting plane/instrument. Given a cut, we only have to evaluate a six digit code, where every digit encodes if the corresponding cell edge is cut or not. The code than allows to access the entry of the lookup table which stores the corresponding decomposition (see the different split configurations in Figure 1). This method allows a reduced fragmentation and is very efficient in time.
We have fixed the lookup table presented in [9] by ensuring the triangulation encoded to be always consistent, i.e. the adjacent split sections should be triangulated such that triangular faces match. As an example, see the configuration showed in Figure 2.a. We have to guarantee that the faces shared by the tetrahedra after the split are triangulated in a consistent manner (see Figure 2.b). More precisely: each quadrilateral face, e.g. p_0, p_1, a, b in the Figure 2 is triangulated with the diagonal p_0, a iff either p_0 or b is the minimum of the four vertices in lexicographical order. Consequently, each configuration possesses a series of subcases determined by the order among the vertices.
Moreover, a special case has to be considered (see Figure 3). We may have three edge intersections which split the tetrahedron in two pieces (one tetrahedron and one prism). It may happen that the lexicographics order of the vertices generates a tesselation of the surface of the prism such that no consistent tetrahedral decomposition of the prism exists (see Figure 3.a). We have to detect when this case arise; the addi-

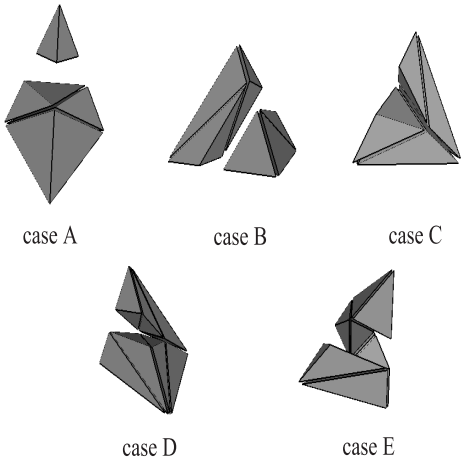


Figure 1: *The different split configurations encoded in the splitting lookup table; the cell is completely cut in cases A and B, and only partially in cases C, D and E.*

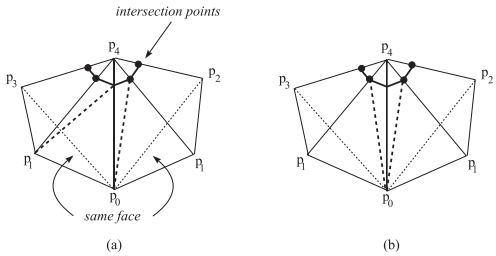


Figure 2: *Two tetrahedra sharing a face; consistency is not preserved in (a), and is preserved in (b).*

tion of a further vertex in the middle of prism allows an easy consistent decomposition (see Figure 3.b).

6. Finding the tetrahedra cut

In this section we explain how we model the interaction between the scalpel and the object. The goal of this step is to find all the intersections between the sweeping surface and the tetrahedra in an efficient way. We distinguish two states:

- the scalpel is in the scene but does not touch any object and
- the scalpel penetrates an object

We use $tail_t$ and $head_t$ to denote, respectively, the positions of the tail and of the head of the scalpel blade at time t. When the scalpel moves to the new position $tail_{t+\delta t}, head_{t+\delta t}$ the sweeping surface is defined by the two triangles $\{tail_t, head_t, tail_{t+\delta t}\}$ and $\{tail_{t+\delta t}, head_t, head_{t+\delta t}\}$ that are assumed to be quasi coplanar.

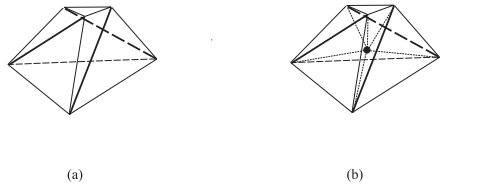


Figure 3: *The tetrahedra section in (a) cannot be decomposed in a manner consistent with the given diagonals (bold lines); we show in (b) a consistent triangulation obtained by introducing an internal vertex.*

6.1. Detecting the initial scalpel-object collision

A very simple and efficient approach is described in [12] to detect collisions between a simple object and a set of geometric primitives. This approach relies on OpenGl features and graphics hardware efficiency. Collisions are detected by defining a *viewing volume* that tightly envelopes the surgery tool. The objects in the surgery scene are then rendered, under the *picking mode* provided by OpenGL (GL_SELECT or GL_FEEDBACK) and therefore the cells that come in contact with the tool are only the ones that are rendered.

This solution provides a very fast and easy to implement solution to detect collisions with the boundary surface of the objects in the scene, provided that a tight view volume can be associated to the surgery tool. We therefore adopted this approach. The active volume swept by the scalpel in two consecutive time steps is represented by a bounding box (see Figure 4), calculated as the smallest box containing the points $\{tail_t, head_t, tail_{t+dt}, head_{t+dt}\}$ [10]. The size of the box edges is slightly larger than the tight enclosing box, to avoid a null volume when the four points are coplanar.
The only disadvantage of this technique is that, if the speed of the scalpel is great with respect to the time step and the size of the objects in the scene, we might trespass the object in a single time step. Moreover, if we have multiple tools, performance can degrade because of the multiple rendering passes needed.

6.2. Moving the scalpel inside the volume

If we use the technique above on a rather complex object (i.e. if we render all faces of the tetrahedral decomposition to find the ones actually touched by the scalpel) we usually get a dramatic degradation of performance. But we can easily detect the internal cells intersected by the swept surface S by propagation from the cells hit on the boundary surface. Following the notation introduced in [1], we call *active tetrahedra* at time t, denoted with A_t, the set of tetrahedra that are intersected by the scalpel. Moreover, $C_{t,t+1}$ are the cells cut in the time interval $(t, t+1)$. At initialization time, A_t is the result of the boundary collision test described in the previous

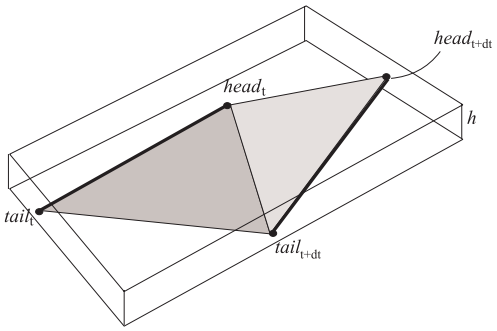


Figure 4: *The surface swept by the scalpel cutting edge in an atomic δt interval of time and a possible corresponding viewing volume are shown.*

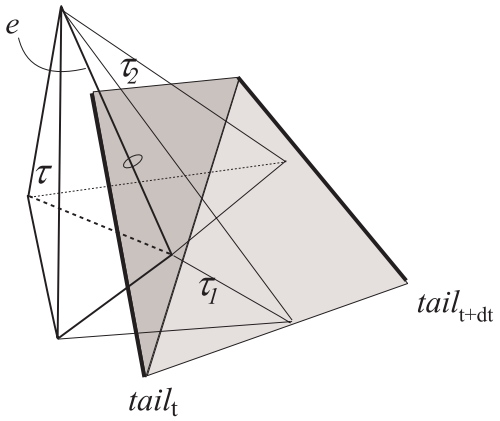


Figure 5: *The sweeping surface intersects the edge e of τ, hence the tetra τ_1 and τ_2 are added to A_{t+1} (and will be tested for further intersections)*

subsection. Then, we proceed by propagation from the cells in A_t. This propagation process allows us to avoid a global search over the complete mesh. The propagation process is as follows:

FindTetraToCut (in A_t; out $C_{t,t+1}$, A_{t+1})
begin
  $C_{t,t+1}, A_{t+1} := \emptyset$;
  while $A_t \neq \emptyset$
    $\tau = Pop(A_t)$;
    if Intersect(S, τ)
      then
      $C_{t,t+1} := C_{t,t+1} + \tau$;
      for each edge $e \in \tau$ such that Intersect(S, e)
          select the two cells adiacent to τ on edge
          e and add them to A_t;
      if the scalpel intersects τ at time $t+1$
        then $A_{t+1} := A_{t+1} + \tau$;
  endWhile
end

The pseudo code above reports only the overall structure of the propagation process; speedup heuristics are not described for clarity of exposition (e.g. when an edge is tested it is also marked, to avoid to execute multiple intersection tests on the same edge).

7. Lacerating the Object

A tissue laceration occurs when an excessive stress is present on some section of the soft object. In our model the stress is directly represented by the elongation of the springs. Therefore the obvious manner to implement lacerations is to assign to each spring a limit on elongation, and to impose that it breaks down when such limit is reached. Since we know how to split tetrahedra, a naive solution would be to mark all the springs that have exceeded their maximum elongation, and then to split all tetrahedra having at least a marked edges/spring. Two problems arise:

- the marked edges could correspond to a not canonical splitting configuration (e.g. when all the edges on a face are marked);
- the too much elongated springs could be disposed so that they produce an unrealistic laceration.)

These problems can be avoided by propagation a valid laceration, which will be called *laceration surface*. When the laceration surface has been defined, we can trivially use the same splitting rule defined to perform cut actions.
First of all, we impose a limitation: any laceration should start and propagate from the surface. In this way the behaviour of the laceration is sufficiently realistic and we reduce the amount of springs to be checked to the ones lying on the surface.
At each time step, an heap containing the subset of springs that are considerably elongated is maintained. The order between elongated edges is given by comparing the ratios between the current elongation and the maximum elongation. The root of the heap contains the most elongated spring, according to the criterion above. When the heap is not empty, a laceration has to be performed. A reasonable criterion can be introduced: we defer to start a new laceration process until the heap size (i.e. the number of too much elongated springs) reaches a threshold value h. In other words, we do not start to lacerate the object until a "sufficient" number of elongated springs are detected. In the following, we describe a simple and intuitive way for propagating laceration within our model.
The laceration surface is defined by propagating a chain of edges $LC = (e_0, e_1, \ldots e_n)$ such that:

- e_0 and e_n are edges on the surface of the soft object;

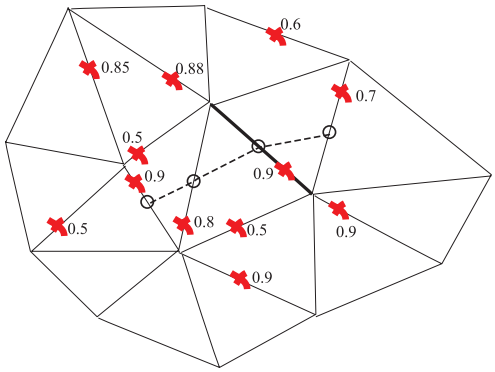


Figure 6: *A step of the**Phase1** in the laceration process: e_0 is the bold edge; all the edges marked are over stressed, and the numbers are the stress values.*

- all pairs of consecutive edges e_i and e_{i+1} are adjacent to a face of the mesh;
- no more than 2 edges in the path belong to the same face.

Figure 6 shows an example of a initial path that respects the conditions above.
The process is subdivided in two phases:

1. find the initial laceration contour on the surface of the deformable object;
2. propagate *LC* through the interior of the mesh.

The first edge of the path *LC* is the edge e_0 found on top of the heap. The first phase corresponds to the invocation of the procedures **RightPropOnSurface** and the corresponding **LeftPropOnSurface**, both starting from the edge e_0. For conciseness reasons, only the first one is described:

RightPropOnSurface (in e_0, out *LC*)
begin
given the surface face f_r right-adjacent to e_0:
choose the edge e of f_r with max elongation;
if elongation(e) > threshold
then
LC:= LC + f_r
RightPropOnSurface (e,LC);
end

Figure 6 shows a step of the algorithm.

A path *LC* has an associated path of faces $LC_{faces} = (f_0, \ldots f_{n-1})$, where f_i is the face shared by edges e_i, e_{i+1}. Furthermore, we use a set of tetrahedra LC_{tetra} initialized as the set of tetrahedra to which the faces in LC_{faces} belongs to.
To propagate the laceration front in the interior of the soft object, we iterate on faces f_i in LC_{faces} the following procedure:

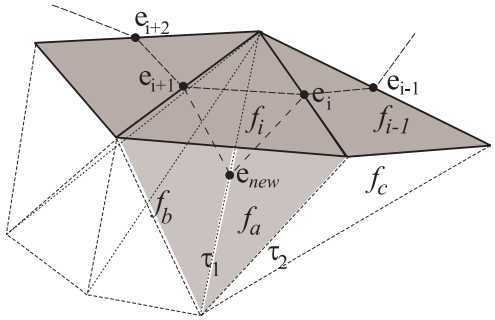


Figure 7: *An example of a step of the procedure **DigFace**: LC_{faces} is updated by removing face f_i and inserting f_a and f_b in LC_{faces} and tetra τ_1 in LC_{tetra}. Then, because face f_a and f_{i-1} are faces of the same tetrahedron τ_2, the front is again propagated ($LC_{faces} = LC_{faces} - f_{i-1} - f_a + f_c$, $LC_{tetra} += \tau_2$).*

DigFace (in LC_{faces}, out LC_{tetra})
while $LC_{faces} \neq \emptyset$
f=Pop(LC_{faces})
given the tetra cell τ associated to the oriented face f,
choose the edge e of τ with max elongation (with e not in f);
if *elongation*(e) > threshold
then
$LC_{tetra} := LC_{tetra} + \tau$;
$LC_{faces} := LC_{faces} - f + IncidFaces(e, \tau)$;
for each $f_j \in IncidFaces(e, \tau)$
if there exist a face f' in LC_{faces} adjacent to f_j on edge e' of a tetrahedron τ' **then**
$LC_{tetra} := LC_{tetra} + \tau'$;
$LC_{faces} := LC_{faces} - f_j - f'$
$LC_{faces} := LC_{faces} +$
$+ \{f'' \mid f'' \in \tau', incident(f'', e)\}$
endWhile
end

The algorithm stops when no faces in the current list LC_{faces} can be further expanded. Note that when a tetrahedron is added to LC_{tetra} three of four of its faces have been in LC_{faces} and it can be split (using the decomposition rules A and B in Figure 1). Furthermore, the tetrahedra that have one or two faces on LC_{faces} but that are not in LC_{tetra} can be partially split (case C,D or E showed in in Figure 1).
A time-critical version of the algorithm can be provided by stopping the propagation after a fixed number of digging operations. In this case we impose to process the faces in sequence through the path LC_{faces} to have a uniform laceration in depth.

8. Results

As stated in Section 5.1, the update of the MT can be performed separately from the update of the current LOD representation used when the cut is performed, with the constraint that resolution remains fixed in the meantime. In this way, time-critical implementation is possible. For this reason, we distinguish, in the tests performed, the time spent in updating the tetrahedra in the current LOD from the time spent in updating the whole MT.
Figure 8 shows a square shaped object made of 600 tetrahedra, hanged on two little cubes that are pulling outward in opposite directions and partially cut by the scalpel. Figure 9 shows that the trajectory of the scalpel can self intersect to draw a hole in the object.
A meaningful measure of performance is the average time spent in the update of the mesh after a scalpel movement in a single step, because it affects the smoothness of motion. Table 1 shows some numerical evaluation of this experiment with an increasing mesh sizes. The more interesting data is the number of tetrahedra that can be cut in a single time step without degrading the performance. As you can see, the average time for updating the current LOD mesh (UpdT column in Table 1) is under 15 milliseconds for an average of 130 tetrahedra cut. Times are expressed in milliseconds and refer to a Pentium II 400Mhz 516MB RAM.
Figure 10 shows a partial and a complete cut on a multiresolution model of the kidney and Table 2 presents the relative results. The update of the DAG is made at the end of the cut process. Note that the tetrahedra cut are not only the ones contained in the LOD representation used at the time of the cut, but all the ones involved in the MT structure.
The frame rate is around 10Hz for representations using under 600 tetrahedra when no cut is performed. In case of cut, the performance can degrade if the cut is performed with a fast movement that causes many tetrahedra to be cut in one step. The time for updating the DAG, shown in Table 2, could affect the performance of the overall process as well, but since this operation is performed only at the end of the cut, it results only in a little transient reduction of the frame rate, and only for cut involving hundreds of fragment and thousand of tetrahedra.

# MS	CuT	CrT	UpdT	Rn T	Rx T
60	1,2	4,7	15	15	19
60	20	113	15	15	15
600	0,8	4,5	15	15	20
600	30	133	23	10	16
6000	24	93	60	40	100
6000	130	586	270	33	80

Table 1: *All the column except the first one report average values (MS = mesh size, CuT = cut tetra, CrT = created tetra, UpdT = time for mesh updating, RnT = rendering time, RxT = time for relaxation). All times are in milliseconds.*

SF	CuT	CrT	TetUp	MT-DAGUpd
16	302	1019	63	109
91	732	4107	250	147

Table 2: *The table presents some data on the two cuts shown in Figure 10 (SF = number of split fragments, CutTetra = cut tetrahedra, CrTetra = created tetrahedra, TetUp = time for update tetrahedra, MT-DAGUpd = time for updating the MT DAG). All times are in milliseconds.*

9. Conclusions and Future works

We have presented the overall organization of a soft object simulation system, which applies multiresolution techniques to support efficiently sophisticated interactions with the soft tissue, including topological modifications such as cuts and lacerations. The implementation of topology modifications on a multiresolution structure (definition of the cut/laceration contour, update of the multiresolution data structure) introduces some time overhead, proportional to the extent of the topological modification. But because we can focus on the data, the reduced resolution of all the sections of the soft object that are only marginally affected by the current manipulation is worthwhile the limited overhead introduced. Moreover, the topology-update procedures have been designed to be interruptible and hence to support time-critical applications.
As stated in the introduction, this work is mostly oriented to the topological and geometrical aspects of soft objects modeling. The physical model adopted in our protypal system is a classic particle system modeling only linear elasticity, but the prototype has been designed in a extensible manner in order to allow further experimentation of the multiresolution approach on other physical models.
In our experiments, the simplification process is performed by an iterative decimation of the mesh, which does not take into account the physical properties of the material. This is correct under the assumption that the Young's modulus would be constant throughout the object. If conversely one has to model non uniform materials, the algorithm used for decimation should be extended to take into account the error produced when sub-volumes with a notable variation (or discontinuity) of E are represented with different triangulations. A material-oriented error evaluation heuristic is therefore needed. One possible solution is to consider the values of E as a scalar field attached to the mesh, and to adopt the common heuristic used for volume dataset simplification [3].

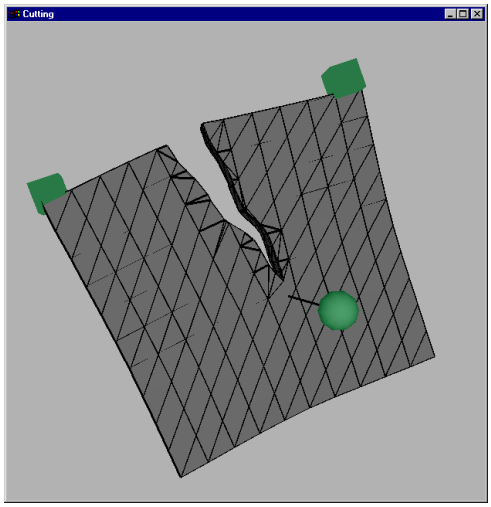


Figure 8: *(a) A square-shaped object cut while two forces pull in opposite directions (mesh rendered in flat-wireframe mode).*

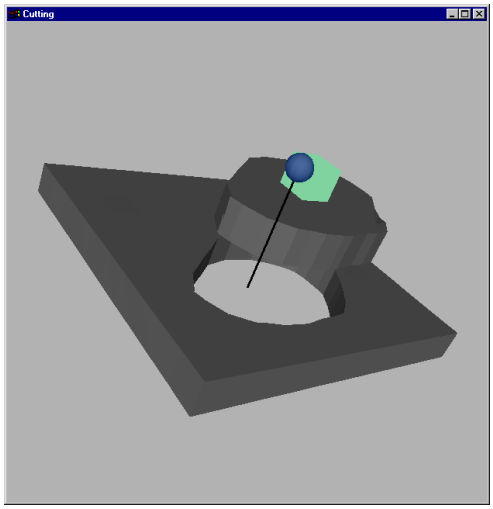
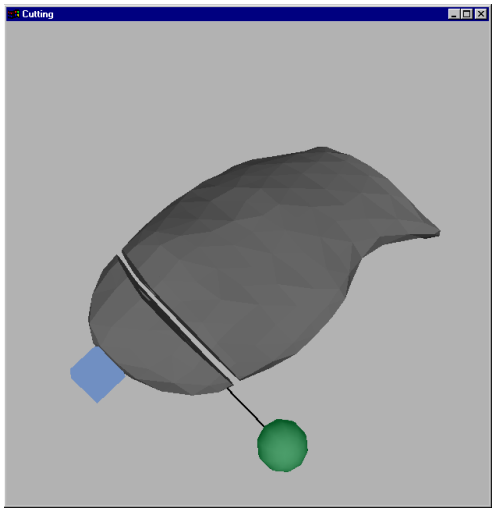
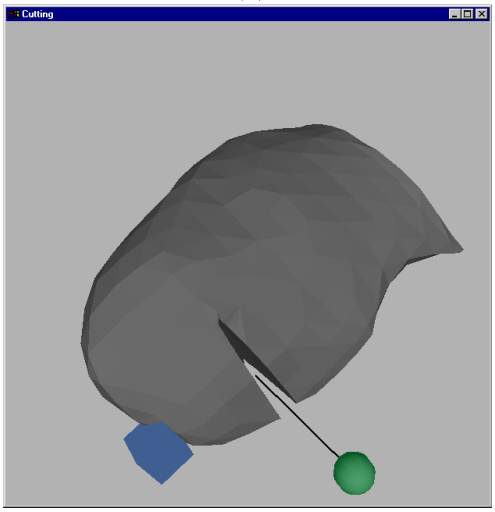


Figure 9: *The trajectory of scalpel can self intersect: scalpel penetrates the square and makes a hole.*

The definition of the cut surface as the intersection between the soft object and the sweeping surface could be considered as too much geometry-oriented approach; it does not allow to model the scalpel as a three dimensional entity, or the introduction of physical interactions, e.g. due to the force feedback of other tissue in the proximity of the cut. To cover these shortcomings, a different model for the definition of the cut is currently under development. Note that this does not affect the rest of the framework, because we only have to introduce a new methodology for the selection of the cells intersected and cut by the surgery tool.



(a)



(b)

Figure 10: *An example of a cut on the kidney multiresolution model: (a) a complete cut; (b) a partial cut.*

Finally, the model does not include any collision detection strategy among soft objects. We only consider the collision between the surgery tool and the soft objects. Considering also inter-objects collision increases substantially the complexity of the problem, because the associated triangulated meshes:

- have not a fixed shape,
- have not a fixed topology, hence no preprocessing phase depending on topology is admitted,
- can be often very close to each one (e.g. the two surfaces on the opposite sides of cut surface).

In our opinion, this could be a stimulating framework for the

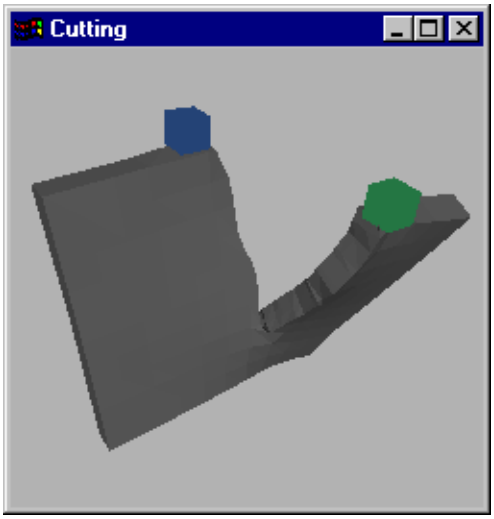


Figure 11: *The little cubes pulling in opposite directions cause a laceration.*

design of new collision detection techniques and it will be also one of our research directions in the next future.

References

1. D. Bielser, V.A. Maiwald, and M.H. Gross, *Interactive cuts through 3-dimensional soft tissue*, Computer Graphics Forum (Eurographics'99 Proc.) **18** (1999), no. 3, C31–C38.
2. 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.
3. P. Cignoni, C. Montani, E. Puppo, and R. Scopigno, *Multiresolution modeling and visualization of volume data*, IEEE Trans. on Visualization and Comp. Graph. **3** (1997), no. 4, 352–369.
4. H. Delingette S. Cotin and N. Ayache, *A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation*, CAS99 Proceedings, May 1999, pp. 70–81.
5. Stephane Cotin, Herve Delingette, and Nicholas Ayache, *Real-time elastic deformations of soft tissues for surgery simulation*, IEEE Transactions on Visualization and Computer Graphics **5** (1999), no. 1, 62–73.
6. 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.
7. G. Debunne, M. Desbrun, A. Barr, and M.P. Cani, *Interactive multiresolution animation of deformable models*, Eurographics Workshop on Computer Animation and Simulation '99 (N.Magnenat-Thalmann and D.Thalmann, eds.), Springer-Verlag, September 1999, pp. 133–144.
8. P. Cignoni F. Ganovelli and R. Scopigno, *Introducing multiresolution representation in deformable modeling*, SCCG '99 Conference Proceedings, Budmerice (Slovakia) (Jiri Zara, ed.), April 28th-May 1st 1999, pp. 149–158.
9. F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno, *Enabling cuts in multiresolution representation*, CGI 2000 Proceedings (N.Magnenat-Thalmann and D.Thalmann, eds.), 2000, p. (in press).
10. Stefan Gottschalk, Ming Lin, and Dinesh Manocha, *OBBTree: A hierarchical structure for rapid interference detection*, SIGGRAPH 96 Conference Proceedings (Holly Rushmeier, ed.), Annual Conference Series, ACM SIGGRAPH, Addison Wesley, August 1996, held in New Orleans, Louisiana, 04-09 August 1996, pp. 171–180.
11. Erwin Keeve, Sabine Girod, Paula Pfeifle, and Bernd Girod, *Anatomy-based facial tissue modeling using the finite element method*, IEEE Visualization '96, IEEE, October 1996, ISBN 0-89791-864-9.
12. J.C. Lombardo, M.P.Gascuel, and F.Neyret, *Real-time collision detection for virtual surgery*, Proceedings of Computer Animation '99, May 1999, pp. 33–39.
13. L. Palazzi and D.R. Forsey, *A multilevel approach to surface response in dynamically deformable models*, Tech. Report TR-94-35, Department of Computer Science, University of British Columbia, January 31th 1994.
14. E. Puppo, *Variable resolution terrain surfaces*, Proceedings Eight Canadian Conference on Computational Geometry, Ottawa, Canada, August 12-15 1996, pp. 202–210.
15. W. T. Reeves, *Particle systems – A technique for modeling a class of fuzzy objects*, Computer Graphics (SIGGRAPH '83 Proceedings) **17** (1983), no. 3, 359–376.
16. S. H. Martin Roth, Markus Hans Gross, Silvio Turello, and Friedrich Robert Carls, *A Bernstein-Bèzier based approach to soft tissue simulation*, Computer Graphics Forum **17** (1998), no. 3, 285–302.
17. O. G. Staadt and M.H. Gross, *Progressive tetrahedralizations*, IEEE Visualization '98 Conf.), 1998, pp. ??–??
18. A. Van Gelder, *Approximate simulation of elastic membranes by triangulated spring meshes*, Journal of Graphics Tools **3** (1998), no. 2, 21–41.
19. O. C. Zienkiewicz, *The finite element method in structural and continuum mechanics*, McGraw-Hill, 1967.