Dynamic Simulation and 3D Interaction

C. Mendoza, C.Laugier & O. Galizzi, F. Faure
SHARP EVASION

INRIA[*] Rhône-Alpes & GRAVIR[†]
ZIRST-655, Avenue de l'Europe, 38330 Montbonnot Saint Martin, FRANCE.
Email: [Cesar.Mendoza-Serrano, Olivier.Galizzi, Francois.Faure, Christian.Laugier]@inrialpes.fr

Abstract

This paper presents some problems and solutions in the domain of dynamic simulation and 3D interaction in virtual reality. First, we handle the trade-off problem between precision and interactivity for the dynamic simulation of a large number of moving solid bodies. Next, in the framework of deformable object interactions we introduce a 3D cutting algorithm including force feedback.

1 Introduction

The dynamic simulation of complex objects and their associated interactions in virtual environments are the origin of many research problematics in robotics and virtual reality. The recent advances in computer science and the use of more powerful machines allows to consider the development of new applications that were unimaginable in the past (e.g. medical simulators). The main problem arises from the need of merging the simulation of physical complex phenomena that combines movements, deformations, interactions (collisions, ruptures, cuts ...) and the resolution of the differential equations representing the behavior of the objects. This problem is constrained further in some applications where interactivity is required (e.g. surgery simulations, videogames).

Developing methods able to trade-off accuracy for interactivity is essential. These methods would allow an application to handle complex scenes and precisely dispatch computation time over the different tasks such as physical simulation, high-level control and graphical rendering. An application interface would include a slider allowing the user to smoothly tune the trade-off between accuracy and frame rate. Currently available methods do not allow the user to limit the computation time, or fail to smoothly degenerate when the computation time decreases, leading to unrealistic results. As the first contribution of this paper, we propose a tunable method for the resolution of the dynamics of rigid bodies and the collision response. This is based on a new efficient Linear Complementary Problem (LCP) solver, with tunable computation time. Error accumulation is avoided by maintaining acceleration, velocity and position errors within acceptable bounds.

The simulated virtual objects can also be deformable (e.g. human organs). To simulate these objects in real-time is difficult due to the complexity of the differential equations representing their dynamics. In some applications, these objects may be subject to different forms of interactions with a human operator. For example, the object might be cut or touched (using a force feedback device). The second contribution of this paper presents an algorithm that allows to execute cuts in soft tissue in real-time. The cutting algorithm reflects cuts by separating the elements that are in contact with the scalpel. To separate them we take into account the physical interaction between the scalpel and the deformable object. The algorithm includes force feedback to increase the realism of the interaction.

The organization of this paper is as follows. Section 2 discusses previous works. Section 3 presents our tunable method and shows how to use it to build an error-tolerant simulation loop. Section 4 presents the cutting algorithm. Next, in Section 5 we present some results and in Section 6 we give a brief conclusion.

2 Previous works

Generally, physical based simulations involve two main tasks : collision detection and resolution of the dynamic equation of the systems. Tunable collision detection methods have been proposed [1] in the past. We focus in tuning the resolution of the dynamic equations. In particular, our dynamic equations represent a system composed of a large number of solid objects that may be moving and interacting between one another. Impact methods handle collisions one after another [2, 3]. Other approaches handle all contacts simultaneously [4, 5]. Moving solids cannot easily be clustered, and multiresolution methods are not well suited because a set of solids is not a continuous material. Since we cannot simplify the model, we need iterative methods able to terminate as soon as a given number of

[*]Inst. Nat. de Rech. en Informatique et en Automatique.
[†]Lab. d'Informatique GRAphique, VIsion et Robotique.

iterations have been performed. Impact methods propagate collisions until impacts fall below a given threshold. This approach is iterative and is the best-suited when propagation effects are desired. It is known to be stable and computationally efficient only when a small number of bodies interact. Pivoting strategies [4] are more efficient but they do not allow us to control the computation time. Milenkovic's method [5] uses optimization and may thus allow trade-offs, though this point is not discussed. The method relies on sophisticated quadratic programming methods and the computation time does not allow the interactive animation of complex scenes. However good stability is obtained by correcting positions and velocities as well as acceleration at each time step. This motivates our work on a simpler, faster and tunable method.

Concerning real-time interactions of deformable objects, in particular 3D cutting of soft tissue and realistic force feedback, most of the research has been done focusing in medical simulations. Previous works address cutting by *removing* [6] from the simulation the elements that collide with the cutting tool or by *subdividing* [7][8] the colliding elements. Removing elements destroys the material from the virtual organ. In some cases, this is not realistic since the mass of the organ is not preserved. To increase realism, the number of simulated elements is incremented. This might cause a slow down in the simulation. On the other hand, subdivision is more realistic, but the number of simulated elements increases and therefore the simulation is slowed down as well. In our previous works [9], we have started a new approach: *separating the elements* instead of removing or dividing them. The approach does not increment the number of elements during the simulations and preserves the mass of the organ. We implemented it in a 2D mass-spring model. Later, Nienhuys et. al. [10] has used the same idea to approach 3D cutting but without using the physical interaction between the tool and the object and force feedback.

3 The tunable resolution method

3.1 Background and motivation

Mathematically speaking, the dynamics of a rigid object subject to geometrical constraints can be written using the Differential Algebraic Equation [11]:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{x}} &= \mathbf{f_e} + \mathbf{f_i} &(1)\\ \mathbf{g}(\mathbf{x}) &= \mathbf{0} &(2) \end{aligned}$$

where $\ddot{\mathbf{x}} = d\dot{\mathbf{x}}/dt = d^2\mathbf{x}/dt^2$, eq. 1 is Newton's law. Vector $\mathbf{g}(\mathbf{x})$ models constraint errors, vector $\mathbf{f_e}$models the external forces and vector $\mathbf{f_i}$models the internal forces applied to maintain the constraint. Articulated solids can be handled the same way in dimension 6, with eq. 2 representing the joint constraints. Differentiating twice eq. 2 and using Lagrange multipliers [12], we get:

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{x}} &= \mathbf{f_e} + \mathbf{J}^T\lambda &(3)\\ \mathbf{J}\ddot{\mathbf{x}} &= -\frac{\delta(\mathbf{J}\dot{\mathbf{x}})}{\delta q}\dot{\mathbf{x}} &(4) \end{aligned}$$

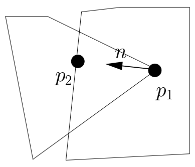
where $\lambda$is the vector of the Lagrange multipliers associated with each independent scalar constraint, matrix $\mathbf{J} = \delta g/\delta q$ is the Jacobian of the constraints, and the second term of eq. 4 depends only on the positions and velocities. This linear equation system can be solved using different techniques[13] with time complexity ranging from $O(n^4)$ to $O(n)$. Linear-time dynamics is restricted to tree-like structures [14, 12], whereas structures with closed loops can be handled using generic cubic-time solutions or quadratic-time iterative solutions based on the conjugate gradient algorithm [15]. Numerical time integration unfortunately results in constraint errors. Moreover, constraint drift necessarily occurs in more complex structures with closed loops. In an articulated solid structure, constraint errors result in broken joints, and penetration at contact points.

A well-known technique to prevent the drift from accumulating over time is to rewrite eq. 4 as $\mathbf{J}\ddot{\mathbf{x}} = -\frac{\delta(\mathbf{J}\dot{\mathbf{x}})}{\delta q}\dot{\mathbf{x}} - \alpha\mathbf{g}(\mathbf{x}) - \beta\dot{\mathbf{g}}(\dot{\mathbf{x}})$. For example, vector $\dot{\mathbf{g}}(\dot{\mathbf{x}})$ may represent the velocity of a body subject to constraints. In practice it is difficult to choose the proportional-derivative coefficients α and β so that the drift remains invisible, unless very small time steps are applied. Moreover the optimal coefficients strongly depend on the simulated structure. Another way of controlling the drift is to apply poststabilization [11]. In this approach, not only $d^2\mathbf{g}/dt^2 = 0$ is enforced at each time step (eq. 4) but also $d\mathbf{g}/dt = 0$ and $\mathbf{g} = 0$. Projecting the state vector to the constraint manifold allows the use of large time steps while easily maintaining constraint drift within acceptable values. In practice, poststabilization cancels geometrical errors due to approximated accelerations as well as numerical integration. We propose a new iterative LCP solver for contact force computation, and show how poststabilization allows us to perform fast approximate computations with efficient control on geometrical errors.

3.2 Iterative solution

We apply constraints to bind solids or to avoid interpenetrations. Figure 1(a) illustrates a frictionless contact constraint set up to avoid interpenetration. Points $\mathbf{p_1}$ and $\mathbf{p_2}$ are computed by a collision detection module. The extraction vector $\mathbf{p_1} - \mathbf{p_2}$ gives the direction $\mathbf{n}$ of a scalar constraint $(\mathbf{p_1} - \mathbf{p_2}).\mathbf{n} = 0$, associated with position error $(\mathbf{p_1} - \mathbf{p_2}).\mathbf{n}$ and required correction $(\mathbf{p_2} - \mathbf{p_1}).\mathbf{n}$. Similarly, the velocity constraint is $(\dot{\mathbf{p_1}} - \dot{\mathbf{p_2}}).\mathbf{n} = 0$ and the acceleration constraint is $(\ddot{\mathbf{p_1}} - \ddot{\mathbf{p_2}}).\mathbf{n} = 0$.

Using extraction vectors to compute the contact constraints allows us to handle polyhedral objects as well



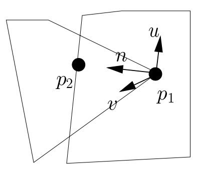


Figure 1: A contact constraint (a) and a point-to-point constraint (b).

as analytical volumes. The object must be convex or decomposed in convex elements. Figure 1(b) illustrates a point-to-point constraint. Points $\mathbf{p_1}$ and $\mathbf{p_2}$ are fixed with respect to their associated bodies. The position constraint $\mathbf{p_1p_2} = \mathbf{0}$ gives three independent scalar constraints along $\mathbf{n}$, $\mathbf{u}$ and $\mathbf{v}$. Velocity and acceleration constraints can be straightforwardly deduced. One way of writing the constrained solid dynamics equation is $\mathbf{JM}^{-1}\mathbf{J}^T\lambda = -\mathbf{e}$ where the sparse matrix $\mathbf{J}$ encodes constraint geometry, the block-diagonal matrix $\mathbf{M}$ encodes masses, λ is a set of Lagrange multipliers (typically the constraint forces along the independent constraint directions) and $\mathbf{e}$ is the error on constrained values which would occur if no constraint forces were applied. $\mathbf{M}^{-1}\mathbf{J}^T\lambda$ is the correction to apply to the accelerations to meet the acceleration constraints. This formulation has several interesting features: constraint solution is seen as the correction of an initial guess, such as null forces or forces computed at the previous time step; matrix sparsity allows the use of a biconjugate gradient algorithm [16], which iteratively refines a global solution even with singular matrices; Lagrange multipliers easily encode a wide variety of constraints [17].

Surface contacts involve inequations since bodies are not allowed to penetrate nor to attract each other. This can be modeled as the following Linear Complementarity Problem [4]:

$$\mathbf{JM}^{-1}\mathbf{J}^T\lambda \geq -\mathbf{e} \quad (5)$$

$$\lambda \geq \mathbf{0} \quad (6)$$

$$\lambda^T(\mathbf{JM}^{-1}\mathbf{J}^T\lambda + \mathbf{e}) = \mathbf{0} \quad (7)$$

where operator $\geq$ applied to vectors means that the inequality holds for all rows. For each contact i, let λ_i be the i-th entry of vector λ (the local contact force) and g_i the i-th entry of vector $\mathbf{JM}^{-1}\mathbf{J}^T\lambda$ (the local acceleration along the normal due to λ). We use an active set method with two sets of contacts: C which contains *clamped* contacts for which we solve for ($g_i = -e_i$, $\lambda_i \geq 0$), and NC which contains *vanishing* contacts for which we set λ_i to 0 and check that $g_i \geq -e_i$. The algorithm iterates to improve an initial guess. At the end of each iteration, contacts in C with attractive force ($\lambda_i < 0$) move to NC whereas contacts in NC where penetration occurs ($g_i < -e_i$) move to C.

Our algorithm is based on the biconjugate gradient solution (BCG). At each time step, it performs two products of matrix $\mathbf{JM}^{-1}\mathbf{J}^T$by vectors. This is done in linear time due to the sparsity of matrices $\mathbf{M}$ and $\mathbf{J}$. Only the subset of $\mathbf{J}$ associated with C is used, since vanishing contacts should not apply forces. When the contact sets are modified, the BCG solution is restarted on the new equation system, with current λ used as initial guess. Since the bodies tend to repell each other, we have never encountered any cyclic behavior so far. Even when the constraints are not feasible (bodies enclosed in a too small space) the penetration converges to a minimum. We approximate Coulomb friction setting additional tangential constraints while clamping the forces inside the friction cones. This is not guaranteed to converge to a correct solution. However, combined with the stabilization presented in section 3.3, this generates visually plausible friction effects.

The important difference between our algorithm and standard LCP solvers [18] is that only one step of the conjugate gradient solution is performed between each inequality checking, whereas most methods perform a full solution using an incremental matrix factoring. Our approach converges faster to the correct sets C and NC. Moreover, the conjugate gradient step is fully applied rather than bounded by the first constraint violation like in pivoting schemes. This prevents the global convergence from being slowed down by a small number of difficult constraints. The number of unknowns (one per independent constraint) is much smaller than using quadratic programming, and force reciprocity is implicitly applied. Consequently, errors can only violate geometric constraints rather than Newton's laws. We compared the efficiency of our algorithm with a standard BCG applied to the same scenes with contacts seen as bilateral constraints. Surprisingly, our algorithm is slightly more efficient in most cases. This is probably due to the vanishing contacts which reduce the number of active constraints.

3.3 Stabilization

High frame rates can be obtained by limiting the computation time spent in the LCP solution. However, we must prevent the consequences of solution error from accumulating at each time step. Fortunately, our LCP solution can straightfordwardly be applied to velocities and positions also. When applied to velocities, $\mathbf{e}$ represents the penetration velocities $\mathbf{J\dot{x}}$ to cancel, and λ represents constraint impulses. Bouncing can be applied using Poisson's model by applying $(1+\epsilon)\lambda$ with $0 \leq \epsilon \leq 1$. Different ϵ can be used at each contact. Applied to the correction of positions, the solution cancels interpenetrations by small displacements. In this case, vector $\mathbf{e}$ represents interpenetration whereas λ represents integrated impulsions. The linear equation holds as long as the displacements are reasonably small. Bouncing and position correction may generate new errors which can be corrected again. Typically, one iteration on velocities and one or two iterations on positions are enough for a visually correct result. Joints may be modeled as inconditionally active constraints. The algorithm may be summarized as follows

```
repeat
   integrate time
   detect collisions
   model collisions and constraint errors
   repeat
      correct positions
   repeat
      correct velocities and apply bouncing
   correct accelerations
   draw
```

4 Soft tissue interactions: 3D cutting with force feedback

In the previous section we have presented some issues concerning simulations with rigid objects. Now, we highlight some contributions in the domain of interactions with soft tissue, such as 3D cutting using force feedback.

4.1 A suitable physical model for deformations and topology changes

We used an explicit formulation of finite element methods [19] to simulate the dynamics of the biological tissue. This model allows topology modifications of the object and it has a strong physical and mathematical foundations. Finite element methods (FEM) partition the object into subelements on which the physical equations are expressed. Instead of merging all these equations in a large matrix system, an *explicit* FEM solves each element independently. It uses the balance equation of each element to obtain the force at each node in function of the displacement of neighbor nodes. Then, instead of obtaining the equilibrium position by solving a large matrix system, we only integrate the force at each node to obtain the new position for the node. We use a non-linear Green strain tensor, ϵ, allowing *large displacements*. The Green strain is expressed by a 3 x 3 matrix. Its (i, j) coefficient is:

$$\epsilon_{ij} = (\frac{\partial \vec{x}}{\partial u_i}\frac{\partial \vec{x}}{\partial u_j} - \delta_{ij}) \tag{8}$$

where the Kronecker delta is $\delta_{ij} = 1$ if $i = j$ or zero otherwise. We assume that our material is isotropic and consider linear elasticity to link stress and strain, as follows:

$$\sigma_{ij}^{(\epsilon)} = \sum_{k=1}^{3} \lambda \epsilon_{kk} \delta_{ij} + 2\mu\epsilon_{ij}. \tag{9}$$

The material's rigidity is determined by the value of μ, and the resistance to changes in volume (dilation) is controlled by λ. The total internal force that a tetrahedron exerts on a node is [19]:

$$\vec{\mathbf{f}}_{[i]}^{el} = -\frac{vol}{2} \sum_{j=1}^{4} \vec{\mathbf{p}}_{[j]} \sum_{k=1}^{3} \sum_{l=1}^{3} \beta_{jl}\beta_{ik}\sigma_{kl} \tag{10}$$

where vol is the volume of the tetrahedron, **p** the position of the nodes of the tetrahedron in the world coordinates and β, the inverse barycentric matrix that links the world positions to the material coordinates. The total internal force acting on the node is obtained by summing the forces exerted by all elements that are attached to the node. Finally, we use a modified-Euler scheme to integrate the dynamics of each node.

4.2 3D volumetric cutting algorithm

Once a collision detection has been detected between the cutting tool and the object, we follow the next steps to carry out cutting phenomena: (1) a geometric criteria, (2) physical criteria, (3) select and separate tetrahedrons, (4) local remeshing and (5) force feedback.

(1) Geometric criteria. We first determine if the user displacements on the surface of the object corresponds to a *cutting attempt* or not. Let $C_p(t)$ be the colliding point at

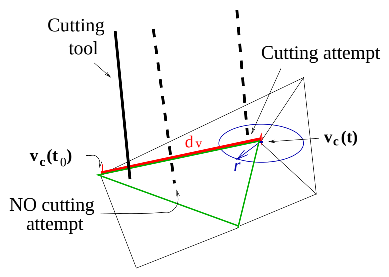


Figure 2: Determining cutting attempts.

time t between the virtual tool and a facet on the surface of the object. Let $v_c(t)$ be the closest vertex to $C_p(t)$ and t_0 the moment of the first contact. Define a neighborhood $\uplus$ around the vertex $v_c(t)$. We consider a *cut attempt* if:

1. $v_c(t) \neq v_c(t_0)$
2. $Cp(t) \in \uplus$

The first condition states that the closest vertex to the colliding points at times t and t_0 must be different. The second condition avoids degenerated cuts due to small movements (e.g. a very small displacement of the tool in the middle of the facet may satisfy the first condition). To cut, the user is forced to execute larger displacements by constraining the tool to lie on the neighborhood $\uplus$, see figure 2. For simplicity, we have chosen the neighborhood to be a circular region with radius r. The value of r determines the size of $\uplus$. Since the facets of a mesh are, in general, of different sizes and forms, the value of r must be computed as a function of the size of the current colliding facet. Thus,

$$r = \alpha d_v \tag{11}$$

where d_v is the distance between $v_c(t)$ and $v_c(t_0)$, see figure 2. This distance changes depending on the colliding facet. The parameter α determines the size of the neighborhood.

(2) Physical criteria. A cut attempt is not enough to break apart the object. Some physical aspects, that take into account the physical interaction between the object and the cutting tool, have to considered. To do that, we consider the internal behavior of the object when it is subjected to external loads produced by the tool. According to fracture mechanics, an object may be broken due to two different type of failures: (a) *Tensile failure:* This corresponds to loading *normal* to the failure surface. If the failure is produced by pushing rather than pulling then we can have a *compressive failure*. (b) *Shear failure:* This corresponds to loading *tangential* to the failure surface. We analyze the forces that cause these failures.

First, note that during the contact, the internal forces equilibrate the external load produced by the tool. The internal force distribution can be represented by an equivalent set of resultants, F, and moments, M, see figure 3. From classical mechanics, the traction, Tr, provides a measure of the direction and intensity of the loading at a given point and it is defined as:

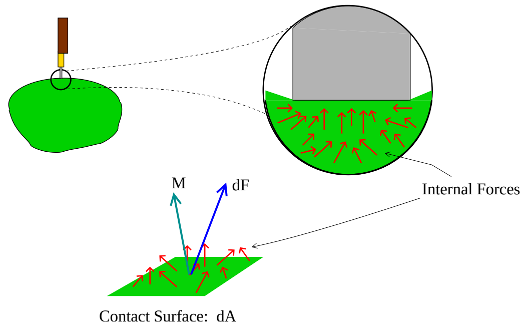


Figure 3: Internal forces during contact between cutting tool and object

$$Tr = \lim_{dA \to 0} \frac{dF}{dA}. \tag{12}$$

Decomposing the force into normal and tangential components, see figure 3, and introducing a sharpness factor, κ, for the cutting tool, we have a *cutting traction* vector:

$$Tc = \frac{1}{\kappa}\left[(\lim_{dA \to 0} \frac{|F_n|}{dA})\vec{n}_1 + (\lim_{dA \to 0} \frac{|F_t|}{dA})\vec{n}_2\right] \tag{13}$$

where n_1 is the normal to the plane and n_2 is the tangent to that same plane,(i.e. a normal in another perpendicular plane). Most of the measurable parameters available in the literature are given using the *fracture toughness*, K_I, of the material which is the critical stress intensity required to produce a failure in a material. Therefore, we put the *cutting traction vector*, T_c, in terms of the stress:

$$T_c = \frac{1}{\kappa}(\sigma\vec{n}_1 + \tau\vec{n}_2). \tag{14}$$

where σ is the *normal* stress and τ the *shear stress*. In the 3D case, T_c takes the following form:

$$\begin{bmatrix} t_{e_x} \\ t_{e_y} \\ t_{e_z} \end{bmatrix} = \frac{1}{\kappa}\begin{bmatrix} \sigma_{e_x x} & \tau_{e_x y} & \tau_{e_x z} \\ \tau_{e_y x} & \sigma_{e_y y} & \tau_{e_y z} \\ \tau_{e_z x} & \tau_{e_z y} & \sigma_{e_z z} \end{bmatrix}\begin{bmatrix} n_{e_x} & n_{e_y} & n_{e_z} \end{bmatrix}. \tag{15}$$

where $\vec{n}_i$ is the normal of each plane of the infinitesimal cube. For simplicity, take Γ as the set of normal and shearing stresses. The object is broken when the maximum stress takes a value greater than the material toughness, K_I. From classical mechanics, the maximum shearing stress is computed using the eigenvalues, σ_1,σ_2 and σ_3 of Γ.

$$\tau_{max} = \frac{1}{2}\mathrm{m}ax\{|\sigma_1 - \sigma_2|, |\sigma_1 - \sigma_3|, |\sigma_2 - \sigma_3|\}. \tag{16}$$

and the maximum normal stress, σ_{max} is the greatest eigenvalue of Γ. Finally, we define our *cutting stress*, σ_c as:

$$\sigma_c = \frac{1}{\zeta} \min{(\sigma_{max}, \tau_{max})}. \tag{17}$$

where $\zeta \in [0.1 \ 1]$ is a parameter representing the *damage* in the cutting area. Finally, a cut is produced if a *cutting attempt* has occurred and if

$$\sigma_c \geq K_I \tag{18}$$

where K_I is the material toughness of the object.

(3) Select and separate tetrahedrons. To select the tetrahedrons which need to be separated to broken the object we consider a *cutting line* on the surface of the object. This cutting line is given by the set of vertices selected as follows: it starts at $v_0 = v_c(t_0)$, the closest vertex to the previous colliding point at t_0, it continues to $v_1 = v_c(t)$, the closest vertex to the current colliding point, such that $v_c(t_0) \neq v_c(t)$. The ending vertex, v_2 of the cutting line is the one that best fits the profile of the cut. We consider that the *cut attempt* is executed in the direction, $\vec{s}_1$, from $C_p(t_0)$ to $C_p(t)$ and that the cut is as straight as possible. We define $\vec{s}_i$ as the vectors from the possible projected vertices to $C_p(t)$; v_2 will be chosen as the vertex v_i whose vector $\vec{s}_i$ is minimum with respect to $\vec{s}_1$:

$$v_2 = v_i \quad \text{such that} \quad \min{\{\angle(\vec{s}_1, \vec{s}_i)\}}. \tag{19}$$

Let $\vec{s}_1$ be the vector from v_0 to v_1 and $\vec{n}$ the normal to the facet as shown in figure 4. Define P as the plane spanned by $\vec{n}$ and $\vec{s}_1$. Set T as the set of tetrahedrons, e^T, sharing the vertex v_1. Then, from e^T, we separate the tetrahedrons, that are in one side of the plane from those that belongs to the other side of the plane, by only splitting v_1. When a tetrahedron, e^T, is divided by the plane, P, the tetrahedron will lie in the side where its furthest vertex lies.

Separating tetrahedrons may create singularities or zero area joints (e.g. tetrahedrons connected only by one vertex). Our data structure, based in a *abstract simplicial complex* K, let us identify these singularities efficiently.

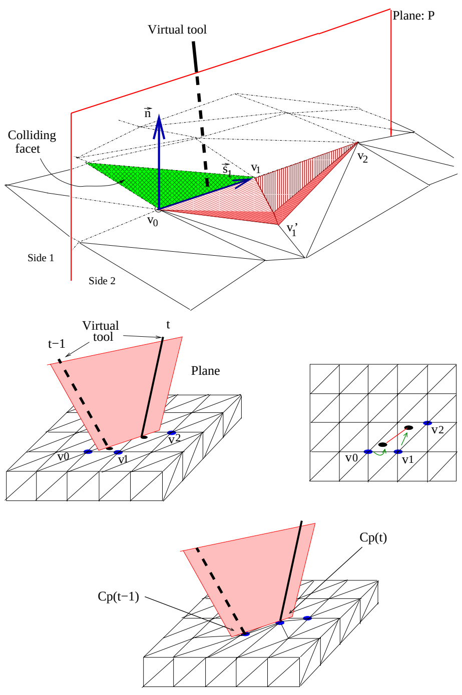


Figure 4: (left)A set of tetrahedra is put in one side of the plane (Side 1) and another set is put in the other (Side 2). The plane is spanned by vectors $\vec{s}_1$ and $\vec{n}$. (right) Repositioning the vertices of the tetrahedrons to fit the cut profile

(4) Local remeshing. To reflect the cut, we reposition the vertices of the cutting line by translating v_0 and v_1 to $C_p(t-1)$ and $C_p(t)$ respectively. The vertex v_2 is not moved until the next cutting step, when it will renamed as v_0, see figure 4. We update the β matrix and the volume of the involved tetrahedrons to keep the physical validity of the model.

(5) Force Feedback. Haptic interaction was included to increase realism. We solve the different rate frequency problem between the physical (about 20 Hz) and the haptic simulations (about 1 KHz) by separating the haptic and the simulation loop and linking them by a *buffer model*. Thus, instead of interacting with the complete model, we interact with a simplified model that allows to compute forces at the haptic frequencies. This buffer model is constructed using a set of tetrahedrons obtained from the object [20]. The computation of the force is obtained using the explicit finite element model on the buffer model. It is possible to reach the haptic frequency using explicit FEM (computations at about 1 KHz.) since the number of tetrahedrons is very small. Note that, in the haptic loop, we do not execute any integration scheme, since the positions of the tetrahedrons are set up by the update of the deformable buffer model. On the other hand, the haptic position, x_{haptic}, is sent to the physical simulation loop.

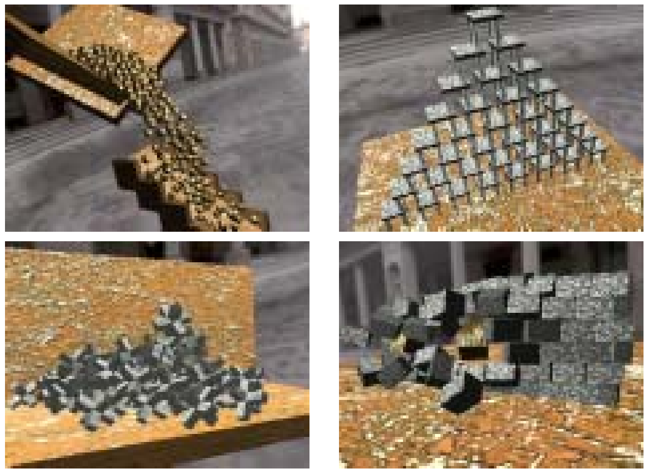


Figure 5: Spheres, stools, jacks, rocks.

	# solids	# contacts/fr	sec/fr
spheres	375	840	0.2
jacks	105	400	0.1
stools	38	141	0.007/0.003
wall	75	190	0.03

Table I: Performances.

5 Results

We have simulated complex scenes with a large number of moving solids, figure 5. Table I summarizes the performance for different scenes. Dynamics computation takes approximately 90 % of the overall computation time. The parameters of our method are the different end loop criteria (precision and number of iterations). The best performance is typically achieved by applying 50% of the computation to position correction, 40% to velocity correction and 10 % to acceleration correction. Crush animations such as *jacks* and *spheres* can be animated using $40ms$ time steps or more, with an overall number of 5 LCP iterations. For *wall* and *stools*, static equilibrium is desired (at the beginning) and a higher number of iterations is required. With 15 iterations at $20ms$, the stools are stable, with 7 iterations they vibrate and with 5 they eventually crush. Compared with recent high quality optimization-based results on similar scenes (*jacks*, *wall*) and taking into account hardware evolution, our method seems to run from 100 to 200 times faster.

In figure 6 we show a physical simulation of an object representing a human knee graft ligament. It is composed of 100 tetrahedrons simulated using explicit finite elements and non-linear Green formalism ($\lambda = 140000$, $\mu = 11000$, $\psi = 10$, $\phi = 80$, sharpness $\kappa = 0.0004$, damage $\zeta = 1$). A PHANToM device is coupled to the simulation to render the sensation of touching and cutting the object. The haptic rendering reaches the 1000 Hz and presents a stable behavior. The haptic sensation of the cut has been largely

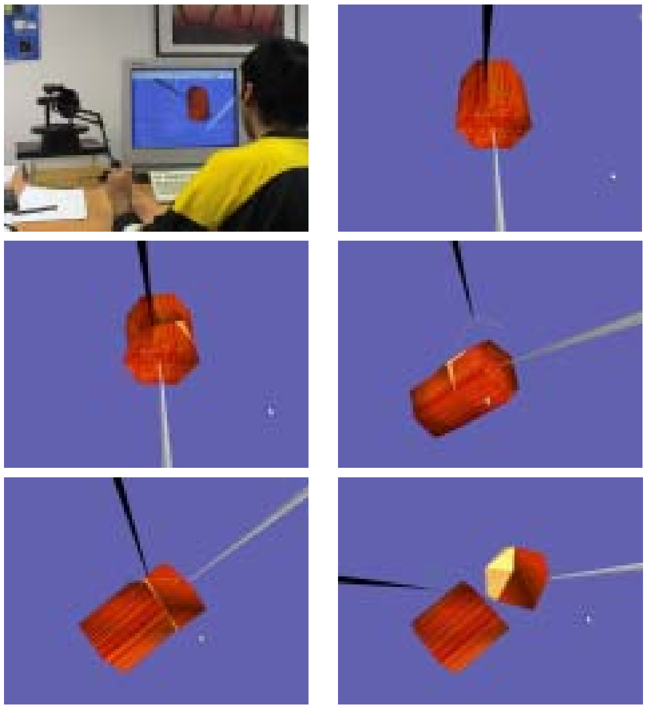


Figure 6: Cutting a human ligament using a force feedback device

influenced by the visual rendering.

6 Conclusion

In this paper we have handled two important issues in the domain of the dynamic simulation and 3D interaction : the trade-off problem between precision and interactivity and 3D cutting of soft tissue including force feedback. For the first, we have proposed a tunable method (Linear Complementary Problem solver) for the resolution of the dynamics and collision response of moving rigid bodies. The method animates several hundreds of bodies in contact at interactive rates. It is especially well-suited for scenes where motion hides small imperfections, such as big crushes. We have also presented a 3D cutting algorithm for soft tissue including force feedback. The algorithm maintains interactivity and preserves the physical realism of the deformable model. The use of a buffer model, constructed from a subset of tetrahedrons of the original model, has allowed force computations at the haptic frequencies.

References

[1] John Dingliana and Carol O'Sullivan, "Graceful degradation of collision handling in physically based animation," in *Computer graphics forum (Eurographics'00)*, 2000, vol. 19, pp. 239–247.

[2] James K. Hahn, "Realistic animation of rigid bodies," in *Computer Graphics (SIGGRAPH '88 Proceedings)*, John Dill, Ed., Aug. 1988, vol. 22, pp. 299–308.

[3] Brian Mirtich and John Canny, "Impulse-based dynamic simulation," in *Proceedings of 1995 Symposium on Interactive 3D Graphics*, 1994.

[4] David Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proceedings of SIGGRAPH '94*, Andrew Glassner, Ed. ACM SIGGRAPH, 1994, pp. 23–34, ACM Press.

[5] Victor J. Milenkovic and Harald Schmidl, "Optimization-based animation," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001, pp. 37–46, ACM Press.

[6] S. Cotin, *Modèles anatomiques déformables en temps-réel.*, Thèse de doctorat, INRIA Sophia Antipolis – Université de Nice, Sophia Antipolis, 1997.

[7] D. Bielser, V. A. Maiwald, and M. H. Gross, "Interactive cuts through 3-dimentional soft tissue," in *EUROGRAPHICS'99*, 1999, vol. 18, pp. C–31–C38.

[8] A. B. Mor and T. Kanade, "Modifying soft tissue models : Progressive cutting with minimal new element creation.," in *MICCAI, Medical Image Computing and Computer Assisted Intervention*, Pittsburg, U.S.A., Oct. 2000, vol. 1.

[9] C. Mendoza, C. Laugier, and F. Boux-de Casson, "Virtual reality cutting phenomena using force feedback for surgery simulations," in *Interactive Medical Image Visualization and Analisys, MICCAI, Holland*, 2001.

[10] H. Nienhuys and F. Vanderstappen, "Combining finite element deformation with cutting for surgery simulations," in *EUROGRAPHICS 2000*, 2000.

[11] Uri Ascher and Linda Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, 1988.

[12] David Baraff, "Linear-time dynamics using lagrange multipliers," in *Computer Graphics (Proceedings of SIGGRAPH '96)*. ACM SIGGRAPH, 1996, pp. 137–146, Addison Wesley.

[13] U. M. Ascher, D. K. Pai, and B. Cloutier, "Forward dynamics, elimination methods, and formulation stiffness in robot simulation," *to appear in International Journal of Robotics Research*, vol. 16, no. 6, Dec. 1997.

[14] R. Featherstone, *Robot Dynamics Algorithms*, Kluwer, 1987.

[15] M. Gleicher, *A Differential Approach to Graphical Manipulation*, Ph.D. thesis, Carnegie Mellon University, 1994.

[16] Press, Teukolski, Vetterling, and Flannery, *Numerical Recipes in C*, Cambridge University Press, 1992.

[17] Paul M. Isaacs and Michael F. Cohen, "Mixed methods for complex kinematic constraints in dynamic figure animation," *The Visual Computer*, vol. 4, no. 6, pp. 296–305, Dec. 1988.

[18] R. Fletcher, *Practical methods of optimization; (2nd ed.)*, Wiley-Interscience, 1987.

[19] James O'Brien and Jessica Hodgins, "Graphical models and animation of brittle fracture," in *SIGGRAPH Conference Proc.*, 1999.

[20] C. Mendoza, "Soft tissue interactive simulations for medical application including 3d cutting and force feedback," *PhD thesis, Institut Politechnique National de Grenoble, INRIA Rhone-Alpes*, May 2003.