HABILITATION A DIRIGER DES RECHERCHES

PRÉSENTÉE À

L'UNIVERSITÉ LILLE I

ECOLE DOCTORALE DE MATHÉMATIQUES

Par Laurent Grisoni

Vers une simulation physique temps réel multi-modèle

Soutenue le 9 Décembre 2005

Après avis de MM. :

Pr. Marie-Paule Cani (INPG/GRAVIR)	.....	Rapporteurs
Pr. Brian Barsky (Univ. Berkeley, USA)	.....	
Pr. Matthias Teschner (Univ. Freiburg, Allemagne)	.....	

Devant la Commission d'Examen formée de MM. :

Pr. Marie-Paule Cani (INPG/GRAVIR)	.....	
Pr. Brian Barsky (Univ. Berkeley, USA)	.....	
Pr. Claude Puech (Univ. Paris Sud, INRIA Futurs)	.....	
Pr. Matthias Teschner (Univ. Freiburg, Allemagne)	.....	
Pr. Philippe Mathieu (LIFL)	.....	
Pr. Christophe Chaillou (LIFL)	.....	Directeur d'HDR

— 2005 —

Résumé

La simulation physique temps réel est un domaine de recherche très actif. La question de la réalisation de simulateurs temps réel touche de manière directe ou non à pas mal de problèmes, par exemple de fidélité mécanique et visuelle, ou de temps de calculs. Pour pouvoir proposer, dans un cadre temps réel, des applications offrant à la fois un bon niveau d'interaction ainsi qu'un niveau correct de simulation, on est souvent contraint à des compromis entre ces divers aspects. La simulation physique est un domaine encore neuf en réalité virtuelle: il n'y a, à notre connaissance, pas de réponse convaincante sur la question de savoir comment proposer des protocoles de manipulation complexes, comprenant, dans un même ensemble, des opérations physiques sophistiquées, telles que la gestion dynamique de liaisons entre systèmes mécaniques, la découpe, la suture d'objets, etc... La littérature propose des solutions pour chaque opération élémentaire, mais la réalisation d'un ensemble complet et souple reste pour le moment très difficile. Pour espérer arriver à un tel résultat, nous pensons qu'une voie intéressante est d'envisager un système mécanique à un niveau conceptuel se plaçant au dessus du modèle, pour, à terme, envisager les systèmes mécaniques comme des "boites noires", aussi indépendantes que possible les une des autres, munis d'une interface leur permettant d'interagir l'un avec l'autre, quand nécessaire. Ces systèmes seraient, avec suffisamment d'indépendance, de surcroît capable d'adapter leur représentation (tant visuelle que mécanique) au contexte d'utilisation, ainsi qu'aux contraintes externes (ex: temps de calcul). Il s'agit de tels objets que nous nommons "multi-modèles". Ce document donne une vue (arbitraire) de la bibliographie traitant de la simulation physique temps réel, et présente nos contributions principales (en les re-situant vis-à-vis de la problématique globale), pour les trois grandes questions que soulève la thématique multi-modèle: premièrement, la maîtrise des modèles et représentations impliqués (pour fournir des simulations aussi précises et rapides que possible), ensuite l'ajout d'adaptivité à ces modèles (permettant d'adapter la compléxité d'un objet), et pour terminer, la question de la co-existence de plusieurs algorithmiques au sein d'un même objet de simulation (pour esperer pallier aux limites inhérentes à chaque modèle).

mots-clefs: réalité virtuelle, simulation physique, temps réel, multirésolution, multi-modèle.

Resume

Real-time physical simulation is, so far, a very active research field. Many different topics are of interest in that field, among which problems dealing with mechanical and visual accuracy, or computation efficiency. Physical simulation remains a very difficult field, where one hardly knows how to propose to users complex manipulation protocols, involving sophisticated physical operations, such as dynamic linking, cutting, using thread for suturing, etc... to the best of the community knowledge, it is possible, for each small part, to propose solutions, but it remains very difficult to propose a whole, global, software solution. In order to try to provide such tools, it is necessary to try to comprehend the problem from higher level than physical model, that is currently, to our knowledge, most of practical research activity in the field: we'd have to find solutions for manipulating, from software point of view, mechanical systems as "black boxes", as independant as possible one from each other, that would provide ways to interact together (e.g. collide). In addition to that, such a view opens the way to a whole new field of non-studied problems, among which the question to know how to provide systems with the ability to dynamically change their representation (either visual or mechanical), depending on the context of use, and external constraints (e.g.: CPU consumption). This is what we call "multi-model" objects. This document presents a (personnal) review of the litterature involved in the field, related to that question, as well as our main contributions, for the main three points involved in the global multi-model question: first, model handling (in order to be able to produce both visually and mechanically accurate objects), second, adaptive models (in a more general way, models that can adapt their complexity for potential automatic tuning), and third, multi-model specific aspects.

keywords: virtual reality, physical simulation, real-time, multiresolution, multi-model.

Il est toujours délicat de remercier de manière complète l'ensemble des personnes qui participent à la mise en place d'un travail tel que celui décrit dans ce document, tant leur nombre est important. Je souhaite en premier lieu remercier les membres du jury pour avoir accepté de donner de leur temps pour évaluer mon travail; Marie-Paule Cani, Matthias Teschner et Brian Barsky, pour avoir accepté d'en être les rapporteurs; Claude Puech et Philippe Mathieu pour avoir participé au jury; Christophe Chaillou, pour m'avoir proposé, au travers de son équipe, un cadre de travail au sein duquel j'ai pu m'épanouir intellectuellement.

Merci à ceux avec qui j'ai eu la chance de travailler ces dernières années a Lille, et qui par leur compétence ont su apporter des éléments essentiels du travail présenté ici, pour renforcer au fil du temps la thématique de ce document: Frédéric Triquet, Julien Lenoir, Jérémie Dequidt, Damien Marchal. Il est très probable que la substance de ce document n'aurait pu se constituer que difficilement sans leur soutien.

Merci à celles et ceux (en priant par avance ceux que j'oublie de me pardonner) qui ont fait, ou qui font de l'equipe GRAPHIX/ALCOVE une équipe extrêmement motivante et soudée, trés agréable à vivre au quotidien: meuh, sylvain, les deux sam, les deux stephane (luigi l'ancien, et le nouveau), nico, fabrice, PJ, sylvere, adrien, lol, et tout ceux que j'oublie...

Je tiens à adresser, avec du recul, un nouveau remerciement à celui qui a été mon directeur de thèse il y a déjà quelques années maintenant, Christophe Schlick: j'avais beaucoup appris de lui en thèse tant scientifiquement qu'humainement, ce que j'avais réalisé il y a déjà longtemps; le modèle humain d'encadrement de la recherche qu'il a su me transmettre participe pour beaucoup à la réussite des différentes activités de recherche que nous avons pu mener ces dernières années (ainsi que celles à venir, souhaitons-le). Je veux qu'il sache, par ces lignes, que je lui suis durablement reconnaissant.

D'une manière plus personnelle, merci également à mes parents, pour m'avoir fait tel que je suis. Aussi à Ugo, pour m'avoir apporté, par sa vision du monde et sa philosophie de vie, une approche nouvelle de la nature humaine.

Je ne veux pas finir cette section sans dédier cette habilitation à Anne, ma femme depuis peu, qui a supporté de manière extraordinaire ma surcharge de travail, mes week-ends studieux de ces derniers temps, et la mauvaise humeur qui en résultait parfois; elle joue un rôle majeur dans l'équilibre global de ma vie. Pour finir, j'envoie un clin d'oeil à la petite puce qui devrait arriver dans notre foyer fin février, et qui regardera, sans nul doute, d'un oeil amusé ces lignes quand elle sera en âge de les lire (en espérant que ce document l'intéressera).

On ne fait jamais attention à ce qui a été fait;
on ne voit que ce qui reste à faire.

Marie Curie

Table des matières

Research Statement (thesis resume, english version) **8**

Introduction **12**

1 Les modèles en simulation physique temps réel **16**

1.1 Modèle géométrique 16

1.1.1 objets linéiques 17

1.1.2 objets surfaciques 20

1.1.3 objets volumiques 21

1.2 Modèle de collision 22

1.2.1 Detection de collision 22

1.2.2 La réponse à la collision 23

1.3 Modèle mécanique 25

1.3.1 lois mécaniques 25

1.3.2 modèles de résolution des équations mécaniques 27

1.3.3 Contraintes 28

1.4 Modèle d'interaction 29

1.5 Contributions 30

1.5.1 Une technique rapide de contrôle de mélange pour représentation implicite [TGMC03] 30

1.5.2 Un modèle de fil B-spline déformable en temps réel[LGM+04] 31

1.6 Conclusion sur les modèles 31

2 Modèles adaptatifs pour la simulation physique temps réel 33

2.1 Un mot sur la terminologie 33

2.2 Modèles géométriques adaptatifs 34

2.2.1 objets linéiques 35

2.2.2 modèles surfaciques 36

2.2.3 modèles volumiques 37

2.3 Modèles de collision adaptatifs 37

2.4 Modèles mécaniques adaptatifs 37

2.4.1 modèle mécanique 38

2.4.2 méthodes d'intégration 38

2.5 Contributions 38

2.5.1 rendu haute performance de cylindre généralisé [GM03] 38

2.5.2 Modèle déformable spline 1D adaptatif[LGMC05] 39

2.5.3 Un modèle volumique déformable à temps constant [DMG05] 40

2.6 Conclusion 40

3 Multi-modèle pour l'animation physique temps réel 41

3.1 Etat de l'art sur le multi-modèle 41

3.1.1 LOD pour l'animation 41

3.1.2 multi-modèle mécanique 42

3.1.3 multi-modèle pour la collision 42

3.2 contributions: une proposition d'architecture adaptée au multi-modèle [DGC04] 42

3.3 Conclusion 43

4 Conclusion et axes de recherches 44

Annexe 1 : CV (English) **57**

Annexe 2: Full publication list **62**

Annexe 3: Relevant publications for the research statement **65**

Research Statement (thesis resume, english version)

Real-time physical simulation has known these last years a growing number of applications. The main original field was scientific calculus; it is now about fields as different as CAD, video games, medical simulation. Each field has its own requirements: CAD demands mechanical correctness, while most video games mostly need plausibility of simulation, i.e. visual mechanical "look alike". Medical simulation tries to reproduce virtually (mostly, so far, for teaching purposes) the behaviour of involved mechanical systems (i.e. organs, muscles, bones, articulations, etc...). The latter field of application is extremely rich, and raises numerous very difficult problems; current solution that are very far from reality. Interaction between organs, the high number of different tissues and topologies, omnipresence of fluids in evolved organism, make the simulation of the whole human system totally unreachable so far. The very best scientific results manage to simulate in real-time one specific organ, along with quite simple interactions (sligh pressure application, simple point catching, simple cut).

The research work presented in this document targets (without any applicative limitations for models involved) medical physical simulation. The ideas presented here are not specific for medical simulation, and we consider them relevant for general real-time physical simulation. Yet, this field provides a very attractive application domain, potentially highly useful to human. The global view of the presented work lies within the decay one can state when, on the one hand, considering the perception human has of the world, and on the other hand, the meaning and limits one can give to the notion of " model ", that is what computer handles for creating animated virtual version of real world. The idea that the perception of an object, in terms of geometry, is hierarchical is quite natural. Such a hierarchy, in the physical simulation context, has to be extended to other domains than geometry in order to gain realism.

A practical example, considered in a natural vocabulary, then re-expressed in the field of virtual reality, makes it possible to apprehend the technological and theoretical problems in which we are interested. Let us consider the case of a tennis ball. Such a ball is made of several layers of rubbery materials, and comprises on its surface thousands of narrowly overlapping fibres. Let us consider this ball as static, at first. Visualizing this ball at very short distance implies to dissociate fibres from/to each other, whereas, while moving a bit away, one has simply the impression of some roughness on the ball; this roughness will turn into a uniform color as one moves away. Let us animate the ball now. Dropping a ball from a low height on the ground generally does not imply, at the scale of human perception, any deformation of the ball. However, the fibres on its surface are actually deformed. When seen from close (and slow) enough, these

deformations will actually appear. If one more strongly pushes on the tennis ball, then the deformation becomes perceptible at average distance. The speed to which the ball moves also influences perception: if the ball is almost motionless and is seen from close, then the details of the ball appear; when visualized at the same distance, but with high speed, the ball will be simply perceived as a yellow sphere.

This small, very simple example ressembles all the key points our research project seeks to analyze: from a computer science point of view, the way one will represent an object, and model it, most of the time very strongly depends on the context in which it will be handled, and visualized. This context of use influences, several point of the object modelling: First, the geometric representation should be able to adapt to the context of visualization (e.g. the ball surface has sometimes to be considered according to the fibre overlap, or simply as a smooth surface). Then, some aspects of the geometry of the object can, according to the context of visualization, to influence on optical modeling (e.g. the fibres induce a rough visual aspect for the ball, saw at intermediate distance, without modifying the spherical geometry). Physical modelling should be able, depending on the context of use, to follow such or such calculation of force, such or such differential equation, use deformable or rigid representation, etc... Each mechanical model having its own domain of application, and its field of validity.

This document presents some results that go towards obtaining adaptive objects, in all the possible meanings of that expression, for physical real-time simulation. At a large scale, these problems imply to carry out in parallel several acquisitions of competence:

- The first thing is to evaluate, as accurately as possible, the various models (either geometric, mechanical, collision, interaction) implied in a simulation, for perceiving their limits in regard of physical simulation. Such a point also implies, from a research point of view, to try to push away, as much as possible, the representation limits, hence improve models.
- Among the potentially useful models, it is necessary to be able to distinguish those that are able to be associated to multiresolution/adaptive methods (from research point of view, it is also interesting, for those that already have this feature, to improve/extend it). This can extend model flexibility, e.g. for tuning memory/computation cost of a given object.
- Third, in order to overcome the intrinsic limits of each representation, and allow practical use of multi-representation objects, one needs to intellectually get one step beyond models, and semantically consider each mechanical system as a whole, closed system: each objet conceptually becomes independent from models, and provides functionalities, instead of representations. This can be seen as a mechanical extension of the notion of level-of-detail, that has been poorly studied in the field of mechanical simulation: yet, we think the major difference lies in the fact that classical LOD considers mostly different data, all together linked to a given functionality (classically display). Here we consider LOD for algorithmes as well. In order to provide convincing solutions, one needs to provide software tools for combining objects in the described way (i.e. with enough functional encapsulation). One also needs to know how to go from a representation to another (e.g. relates to conversion between geometric models), and when (i.e. which elements to extract from context of use, for decision making).

For each of those three points, we proposed first results, which are included in this document as

published article (Appendix 3).

First part: what is the part of reality that models can represent? Such a field, considering it includes geometry, mechanical, collision and interaction, is extremely large, and the existing literature, huge. We present here two contributions to the field, the first one dealing with geometric representation, the second one dealing with mechanical model:

- [TG03] (work achieved with Frédéric Triquet and Philippe Meseure) This work presents a solution for controlling blending for implicits. It describes an efficient way to encode and treat the blending graph, that is necessary to handle when, as in the practical case we were interested in, deforming a squeleton-based implicit: blending is useful for combining narrow primitives together, but has to be prevented when deforming the object again itself.
- [LG04] (work achieved with J. Lenoir during his PhD) This article main contribution is a set of constraints for B-spline based deformable 1D model, but also gives summary of a spline mechanical model we proposed, and published elsewhere [7]. The targeted application is surgical thread: for handling dynamic suture, such constraints as sliding point, i.e. the ability to slide a 1D object through some pre-defined space point, is a mandatory tool. The presented article describes the energy terms that were used,. Lagrangian multipliers are used for constraints solving, that provides more accurate constraint gestion than classical penalty-based correction.

Second part : which models can tune their complexity ? The second part of our work studies which model (either geometric, mechanical, collision management, or interaction) can provide multiresolution, or adaptivity, and aims at proposing efficient and flexible multiresolution/adaptive models. We present here three contributions:

- [GM03] (work achieved with Damien Marchal) This contribution deals with adaptivity of geometrical model, proving high-performance display for generalized cylinders. Apart from the new display technique (that provides smoother extrusion of the section, while drastically improving display performances), it is our very first attempt to provide to some model a functionnal control box, that automatically adapts the model (in that case the display process) to some external performance requirements
- [LM05] (work achieved with J. Lenoir during his PhD) This work extends [LG04] by providing some adaptive version of the spline deformable model. This is associated to dynamic rules, that take both mechanical and geometrical aspect into account, in order to provide adaptivity. This model allows, among others, for knot tying, at arbitrary parameter value, while maintaining the thread resolution almost constant.
- [DM05] (work achieved with Jérémie Dequidt during his PhD, and Damien Marchal) we proposed an adaptive, deformable model, based on finite elements. Both collision and mechanical model are made adaptive. Use of hierarchical structures in both of these models is achieved when possible. The reader may note that the same feature present in [GM03] has been associated to this object, i.e. a tuning algorithm that dynamically adapts the model resolution (both on collision and mechanic) to the external performance requirements. Such tuning tools (both on geometry and simulation) are potentially good

candidates for true guaranteed framerate physical simulation applications (i.e. application that, without any specific knowledge of the calculation power available, can provide some pre-defined framerate, by adapting the model costs when needed).

Third part : how to define multi-model objects in real-time physical simulation ? how to go from a model to another if needed, and when ? This last part deals with the general question to know how to think beyond the model, and only consider each mechanical system as a set of functionnalities. This implies to get enough freedom from models to be able to change from one to another, and decide when appropriate to change. We present here our contribution to that field: a software architecture that encapsulates each mechanical system within an autonomous set (an agent) [DG04] (work achieved with Jérémie Dequidt during his PhD) this work explored in which measure one can provide freedom to mechanical systems within real-time simulation, and provide a software framework for such. Simple cases, illustrating flexibility of the architecture are exhibited, and the potential advantages are discussed.

Introduction

Contexte général

Nous présentons ici une synthèse des travaux menés depuis notre arrivée, en septembre 2000 en tant que maître de conférences attaché pour les activités d'enseignement à l'ecole d'ingénieur Polytech'Lille, dans l'équipe GRAPHIX (aujourd'hui projet INRIA ALCOVE) du LIFL, équipe dirigée par le Pr. Christophe Chaillou. Lors de ma thèse, effectuée à l'université de Bordeaux I sous la direction de Christophe Schlick, j'avais eu l'occasion de m'interesser, dans le contexte de la seule modélisation géométrique, aux représentations splines et implicites, entre autres aux aspects multirésolution par ondelettes sur ces modèles. l'équipe Graphix, à mon arrivée, avait alors deux thématiques centrales: d'une part la simulation physique temps réel, et d'autre part le travail collaborative 3D distant. Assez rapidement mon activité de recherche s'est orientée vers la première thématique, dans laquelle se situe la synthèse ici présentée. J'ai eu la chance de bénéficier, de septembre 2003 à aout 2005, d'une délégation INRIA, qui m'a permis de mûrir le travail présenté dans ce document. Le travail que je présente ici se base entre autres sur les activités d'encadrement de recherche ci-dessous:

- Thèse de Julien Lenoir (soutenue le 23 novembre 2004), Université de Lille 1, *Modèle déformable 1D pour la simulation physique temps réel*: cette thèse a permit de définir un premier modèle spline déformable 1D temps réel disposant d'un ensemble de contraintes lagrangiennes (notamment la contrainte de point glissant), et bénéficiant de propriétés adaptatives, permettant entres autres le serrage d'un noeud à complexité de modèle quasi-constante;
- Thèse de Jérémie Dequidt (date prévue de soutenance: 9 décembre 2005), démarrée en septembre 2003, Université de Lille 1: *Objets autonomes en simulation physique temps réel*: cette thèse rassemble les premiers pas de notre thématique de recherche concernant le multi-modèle proprement dit. Elle s'interesse d'une part au contexte logiciel dont il est souhaitable de disposer pour envisager sereinement de tels objets, et ensuite propose quelques premiers exemples d'objet pouvant adapter leur représentation à des contraintes extérieures.
- Mémoire de DEA de Jérémie Dequidt,*Détection de collision entre objets physiques autonomes*, Université de Lille 1, juillet 2002: ce mémoire de DEA, préambule à la thèse du même étudiant, a posé les bases de l'architecture, en proposant un pipeline de collision suffisament modulaire pour donner une bonne autonomie aux systèmes mécaniques simulés.

Le lecteur pourra se référer à l'annexe 1 pour une liste exhaustive de nos encadrements de recherche (ainsi que de nos diverses activités liées à la recherche).

Sujet de recherche et organisation du document

L'utilisation de l'animation à base physique connait ces dernières années un nombre d'application de plus en plus important: à l'origine reservé au calcul scientifique, la simulation physique concerne maintenant des domaines aussi divers que la CAO, le jeu vidéo, ou la simulation médicale. Ces trois domaines n'ont pas le même rapport à la simulation: le jeu vidéo se contente souvent de *plausibilité* de la simulation vis-à-vis de la réalité; la CAO necessite une réelle adéquation de la simulation avec la réalité; la simulation physique médicale, qui vise à reproduire de manière virtuelle le comportement des tissus organiques, tente, elle de coller, dans la mesure du possible, avec la réalité. Ce dernier domaine d'application est d'ailleurs un domaine extrèmement riche, de par le nombre de problèmes concrets actuellement tres incompletement résolus, et l'extreme complexité mécanique des tissus vivants. L'interaction mécanique entre les organes, le nombre important de tissus et de topologies différents, l'omniprésence des fluides dans un organisme evolué, rendent la simulation physique d'un organisme humain par exemple totalement utopique à l'heure actuelle. Les meilleurs résultats scientifiques arrivent pour le moment à simuler tel ou tel type d'organe, avec, en terme d'interaction, certaines opérations simple (tel l'application d'une legere pression sur cet organe, une préhension simple, ou une découpe primitive).

Le travail présenté dans ce document a pour vocation essentielle la simulation physique médicale, sans pour autant que la philosophie de fond en soit dépendante: les modèles présentés, ainsi que l'idée de fond de ce document sont incontestablement utilisables dans d'autres contextes que la simulation médicale. Toutefois, ce contexte, de par sa complexité, constitue un terrain d'exploration tout à fait propice à l'illustration des quelques principes que ce document ébauche.

Le fil conducteur de ce document repose sur l'examen du décalage que l'on constate en analysant d'une part la vision que l'être humain peut avoir du monde et des objets qui l'entoure, et d'autre part le sens que l'on peut donner à la notion de "modèle", qui est ce que la machine manipule pour créer une image. L'idée que la perception d'un objet, au sens géométrique du terme, se fasse de manière hiérarchique, est assez naturelle. Toutefois, cette hiérarchie doit, dans le contexte de l'animation, être étendue à d'autres domaines que la géométrie pour gagner en réalisme.

Un exemple concret, considéré dans un vocabulaire naturel, puis ensuite recadré dans le contexte de modélisation en réalité virtuelle, permet d'appréhender les verrous technologiques et théoriques auxquels nous nous intéressons ici. Considérons le cas d'une balle de tennis. Celle-ci est constituée de plusieurs couches de matériaux caoutchouteux, et comporte sur sa surface un ensemble de fibres étroitement imbriquées. Considérons la tout d'abord immobile. Visualiser cette balle de prés implique de dissocier les fibres les unes des autres, alors que, en s'éloignant un peu, on aura simplement l'impression d'une certaine rugosité sur la balle, laquelle rugosité disparaîtra pour laisser place à une couleur uniforme au fur et à mesure que l'on s'éloigne. Animons maintenant notre balle. Lancer une balle depuis une faible hauteur sur le sol n'implique généralement pas, à la perception humaine, de déformation de la balle. Pourtant, les fibres à sa surface le sont. Visualisée de prés, ces déformations apparaîtront. Si l'on presse plus fortement sur cette même balle, alors celle-ci se déformera. La vitesse à laquelle évolue la balle influe également sur la perception que nous en avons : si celle-ci est quasiment immobile et visualisée

de prés, alors les détails de la balle apparaîtront, alors que si, visualisée à la même distance, sa vitesse est grande, celle-ci sera simplement perçue par l'œuil humain comme une tâche jaune.

Ce petit exemple simple, aussi basique puisse-t-il être, recèle en lui les points clefs que notre projet de recherche cherche à analyser, à savoir que, du point de vue informatique, la manière dont on va représenter un objet, le modéliser, dépend très fortement du contexte dans lequel il est manipulé, et visualisé. Il influe, du point de vue de la modélisation de l'objet, sur plusieurs facteurs: d'une part, la modélisation géométrique d'un objet doit pouvoir s'adapter au contexte de visualisation (la surface de la balle doit parfois être envisagée selon l'imbrication de fibre, ou simplement une surface lisse). Ensuite, certains aspects de la géométrie de l'objet peuvent, selon le contexte de visualisation, influencer sur la modélisation optique (les fibres introduisent un aspect rugueux pour la balle, vue d'une distance intermédiaire, sans modifier la géométrie sphérique). Quant à la modélisation physique, elle pourra, suivant le type de contraintes, suivre tel ou tel bilan de force, telle ou telle équation différentielle, se déformer ou non, etc... Chaque modèle mécanique ayant son cadre d'application, et son domaine de validité. La co-existence pour un même objet de diverses représentations, et de divers niveaux de représentation au sein d'une même modélisation, est donc une propriété potentiellement attirante. Cette propriété de multirésolution permet d'envisager un objet sous forme de plusieurs "versions", allant des plus grossières aux plus fines. La mise en place de tels outils nécessite toutefois un investissement (notamment logiciel) important. Ce dernier aspect nous amène ainsi à envisager un autre point, nécessaire quant à la mise en place d'objets multirésolution/multireprésentation.

Ce document présente quelques résultats qui vont vers l'obtention d'objets adaptatifs, dans tous les sens du terme, pour la simulation physique temps réel. D'une manière globale, cette problématique implique de mener en parallèle plusieurs acquisitions de compétence: il s'agit tout d'abord de suffisamment maîtriser les divers modèles (tant au niveau géométrique que mécanique, ou pour la détection de collision) impliqués dans une simulation à base physique pour être capable d'en percevoir (éventuellement d'en repousser) les limites. Ensuite, dans les divers modèles utilisables, il faut pouvoir distinguer ceux pouvant être associés à des propriétés multirésolution (éventuellement de les améliorer, ou d'en proposer de nouvelles), ce qui peut donner une certaine souplesse, pour éventuellement adapter la complexité calculatoire d'un objet à un contexte d'utilisation. Enfin, pour dépasser les limites de chaque représentation, et permettre une réelle utilisation d'objets multi-représentation, l'on a besoin de techniques permettant, de manière totalement transparente pour l'application globale, à un objet de changer de représentation (pour telle ou telle partie algorithmique, entre la détection de collision, la mécanique, l'affichage, etc...): précisément parlant, cela implique d'une part la proposition d'un cadre logiciel permettant effectivement une telle séparation suivant les systèmes mécaniques envisagés, et d'autre part l'existence de règles de décision permettant à l'objet, de manière dynamique, d'adapter sa représentation à son contexte d'interaction.

Ce document est donc organisé en trois parties: le premier chapitre traite des modèles (tant géométrique que pour la collision et la mécanique), en tentant (car le domaine est extrêmement large) un tour d'horizon des techniques qui sont utilisées pour les trois classes de représentation étudiées ici. Nos contributions sur ce point sont également résumées:

- Une technique rapide de contrôle de mélange pour représentation implicite (travail réalisé en collaboration avec F. Triquet et P. Meseure);

- Un modèle de fil B-spline déformable en temps réel, associée à un ensemble de contraintes résolues par multiplicateur de Lagrange (travail réalisé avec J. Lenoir dans le cadre de sa thèse).

Le deuxième chapitre de ce mémoire traite plus spécifiquement des modèles, parmi ceux cités dans le premier chapitre, qui peuvent être associés à des techniques adaptatives, afin par exemple de respecter des contraintes externes (occupation mémoire, coût de calcul, etc...). Nos contributions sur ce point sont également replacées dans le contexte bibliographique, elles sont, pour ce chapitre, au nombre de trois:

- Une technique adaptative de rendu de cylindre généralisé, permettant à la fois un rendu de qualité et des performances bien plus hautes que les techniques classiques, pouvant notamment s'adapter à une contrainte de temps d'affichage imposée extérieurement (travail réalisé en collaboration avec D. Marchal);
- Une version adaptative du modèle de B-spline 1D déformable, permettant entre autre le serrage, en temps réel, d'un noeud à un endroit arbitraire de la spline (travail réalisé avec J. Lenoir dans le cadre de sa thèse);
- Un modèle volumique déformable ayant toutes les propriétés adaptatives lui permettant d'adapter (moyennant une certaine tolérance) son temps de calcul d'animation (collision+intégration mécanique) à un temps imposé de manière externe (travail réalisé avec J. Dequidt, dans le cadre de sa thèse, et D. Marchal).

Pour terminer, la troisième et dernière partie de ce mémoire discute de manière plus globale du multi-modèle, pour, au delà des techniques liées aux représentation, ou même au delà du coté multirésolution/adaptatif de certaines, pouvoir s'abstraire d'elles, et permettre à un objet, à un système animé, d'en changer dynamiquement. Après une étude des quelques travaux pouvant être reliés à cette problématique, notre contribution principale à cette problématique est introduite: une architecture logicielle qui permet à un objet de changer une partie de sa représentation sans que cela n'ait de conséquences sur les autres systèmes physiques impliqués dans la simulation (travail réalisé avec J. Dequidt, dans le cadre de sa thèse).

Chapitre 1

Les modèles en simulation physique temps réel

La simulation physique temps réel est un domaine très vaste, où la recherche est à l'heure actuelle très active. Donner une vue d'ensemble exhaustive est une tâche très difficile (le lecteur peut par exemple se référer à [Mes02] pour un excellent survol de ce domaine). La littérature liée au domaine est très large, et des algorithmes, adressant des problématiques très différentes, doivent co-exister au sein d'une application de simulation physique. Nous tentons ici une classification rapide des familles d'outils nécessaires pour créer un simulateur physique, en les regroupant par intérêt fonctionnel, suivant quatre groupes: modèle géométrique (pour l'affichage), détection et gestion des collisions, modèle mécanique proprement dit, interaction utilisateur. Il est à noter que certaines des techniques présentées ici peuvent être modifiée pour manipuler des modèles de manière adaptative: ceci sera traité en détail au prochain chapitre, et par conséquent, aucun aspect multirésolution ou adaptatif ne sera traité ici.

La notion de "modèle", telle qu'utilisée dans ce document, est assez floue: on parlera souvent, indifférent, de "modèle" pour l'affichage, où l'on considère souvent uniquement la géométrie et les différentes représentations que l'on peut en faire, et de "modèles" (par exemple pour la mécanique) qui sont plutôt relatifs à une algorithmique (ex: modèle d'intégration numérique). Dans les cas ambigüs, le terme "modèle" sera dans ce document prioritairement utilisé pour désigner un modèle algorithmique: on utilisera l'expression "géométrie support" pour désigner la représentation géométrique sur laquelle l'algorithmique s'exécute.

1.1 Modèle géométrique

La modélisation géométrique est un domaine très riche, et toutes les problématiques de celle-ci n'ont pas de lien immédiat avec la problématique étudiée dans ce document. Nous donnons ici une vue des outils et modèles, que l'on peut considérer comme venant du monde de la modélisation géométriques, qui sont soit déjà couramment utilisés, soit dont le potentiel, dans le cadre de l'affichage pour la simulation physique temps-réel, nous semble important. Tous les arguments développés ici se font dans ce cadre précis, et dans ce cadre uniquement. Il motive notamment les choix de présentation faits, dans la mesure où les modèles cités ici sont soit en

lien direct avec les représentation couramment utilisées pour la simulation mécanique, soit parce que l'on dispose de techniques de conversion rapides depuis les modèles utilisés par la mécanique, vers ces modèles.

La description que nous avons choisie de développer ici s'organise suivant la dimension des objets représentés. Ceci peut sembler arbitraire, et, comme nous le verrons, certains modèles sont d'ailleurs utilisables pour plusieurs dimensions différentes d'objets (ex: les splines pour les objets 1D, 2D et certains objets 3D): Dans le cadre de la simulation physique, cette classification nous a semblé la plus naturelle, dans la mesure où les arguments de contrôles topologiques (cruciaux pour des opérations comme la découpe ou la déchirure par exemple) sont souvent déterminants dans les choix effectués, et lourdement dépendants, pour un modèle géométrique donné, de la dimension de l'objet à simuler.

1.1.1 objets linéiques

Cette classe de modèle rassemble toutes les simulations d'objet 1D, du fil textile au cheveu, en passant dans le contexte de la simulation chirurgicale par le fil de suture, qui reste à l'heure actuelle un réel challenge du point de vue mécanique (nous aurons l'occasion d'y revenir dans la section 1.5.2. On assimile assez simplement une courbe 1D à une courbe paramétrée, par le biais de deux grandes familles de modèles:

- Les splines: il s'agit de la famille de modèle la plus ancienne, dont les premières formes sont venues en solution à l'interpolation polynomiale par morceaux d'un nombre fini de points [Sch46]. Le cas de la ligne brisée (suite de points reliés entre eux par des segments de droite, le cas le plus basique de formulation linéique continue) peut-être vu comme un cas particulier de B-spline uniforme, de degré 1 (voir ci-dessous). Ces modèles donnent de manière naturelle une paramétrisation de la courbe. Il existe plusieurs modèles de splines [Gri99], mais tous peuvent se définir de la façon suivante: on considère comme courbe spline toute courbe pouvant s'écrire comme combinaison linéaire de fonctions prédéfinies:
$$C(t) = \sum_{k=0}^{n} P_k F_k(t) \tag{1.1}$$
Les familles de splines les plus utilisées à l'heure actuelle, comme les NURBS [PT95] ou les B-splines [Far92] intègrent des propriétés sophistiquées, comme par exemple le contrôle du niveau de continuité paramétrique de la courbe, un contrôle fin de la paramétrisation, via le *vecteur de noeud*. Un vecteur de noeud est, dans la terminologie B-spline, une séquence croissante de valeur:
$$T = \left(\begin{array}{cccc} t_0, & t_1, & \ldots, & t_M \end{array} \right) \qquad \text{avec} \qquad t_0 \leq t_1 \leq \ldots \leq t_M$$
A partir de ce vecteur nodal et d'un degré polynomial, on peut construire les fonction d'influence B-splines à l'aide d'une définition récursive, présentée par Cox [Cox72] et De Boor [dB72] :
$$N_{i,0}(t) = \begin{cases} 1 \,\text{si}\, t_i \leq t \leq t_{i+1} \\ 0 \,\text{sinon} \end{cases}$$
$$\text{et} \tag{1.2}$$
$$\forall\, p \in I\!N^*,\ N_{i,p}(t) = \tfrac{t-t_i}{t_{i+p}-t_i} N_{i,p-1}(t) + \tfrac{t_{i+p+1}-t}{t_{i+p+1}-t_{i+1}} N_{i+1,p-1}(t)$$

Les NURBS peuvent, elle, s'envisager comme une extension rationnelle des courbes B-splines, et beaucoup de leur propriétés découlent directement de celles des B-splines.

- Les courbes à subdivision: celles-ci ont fait leur apparition en informatique graphique il y a environ 15 ans[DZ00]. Ces courbes sont construites itérativement, comme point fixe d'une suite de transformation. En formalisant la subdivision sous la forme d'un produit matriciel, on peut, moyennant l'étude spectrale de la matrice de subdivision impliquée, dégager des méthodes d'evaluation directe des courbes (et surfaces) à subdivision. Cette technique est à la base de la manipulation de la parametrisation de tels objets. Il est à noter toutefois qu'elle nécessite un investissement théorique élevé, et n'est de surcroit pas, du moins à notre connaissance des résultats actuels, manipulable pour tous les schémas de subdivision (la linéarité est en effet au moins nécessaire, ce qui rend par exemple cette technique non-disponible pour des schémas tels [AEV03] ou [HSB05]) .

Bien souvent, l'affichage des modèles 1D nécessite l'association d'une "épaisseur" à la courbe. Ceci se fait la plupart du temps par le biais de deux classes de techniques.

- objets implicites: les objets implicites sont des représentations intrinsèquement volumiques, et définissent la surface d'un objet comme l'ensemble des points solution d'une certaine équation:

$$S = \{P \in \mathbb{R}^3 \ \mid \ F(P) - T = 0\} \tag{1.3}$$

Dans le cas des courbes 1D, les représentations implicites les plus intéressantes sont les objets implicites à squelette: les représentations par convolution [She99], et les représentation equipotentielles. Les deux constructions utilisent pour définir la surface une expression permettant de calculer une valeur scalaire à partir du point de l'espace, et du squelette. Les surfaces de convolution utilisent comme expression une grandeur pouvant etre vue comme l'integrale, sur le domaine du squelette V, d'une expression dépendant d'un *noyau* analytique (le paramètre s contrôle la zone d'influence du noyau):

$$F(P) = \int_V h(P - r)dr \tag{1.4}$$

le noyau h a le plus souvent la structure d'une gaussienne isotrope. Certaines expressions ont l'avantage, dans le cas de primitives de squelettes simples, de permettre une évaluation symbolique du terme intégral. Par exemple, le noyau de Cauchy[She99] fourni une expression symbolique de F sur un squelette formé de points isolés, de segments de droite, d'arcs de cercle, ou de triangles, rendant ainsi accessible une gamme trés large de squelette:

$$h(r) = \frac{1}{(1 + s^2 r^2)^2} \tag{1.5}$$

Le paramètre étant un facteur de contrôle du comportement géométrique de l'objet résultant. Les objets dit *équipotentiels* sont d'une définition analytique plus simple: on les définis généralement par le biais de l'application d'une fonction de mélange (en tout point semblable aux noyaux de convolution cités plus haut) sur la distance entre le point considéré de l'espace, et le squelette:

$$F(P) = h(d(P, V)) \tag{1.6}$$

Ces représentations présentent, dans le cadre de l'affichage, la propriété intéressante de garantir une bonne continuité de surface, quelque soit la courbure locale du squelette sous-jacent: de ce point ils sont plus attractifs que les cylindres généralisés (cf point suivant). Toutefois, ces derniers sont généralement moins coûteux à afficher, dans la mesure ou les objets implicites doivent, de manière générale, passer par une tesselation avant de pouvoir être visualisée [HAC03].

- cylindres généralisés: ils sont une construction classique en modélisation géométrique, et considèrent une surface comme étant générée par extrusion d'une courbe (le plus souvent fermée) le long d'un axe. Cette approche est particulièrement adaptée à la modélisation de tous les objets pour lesquels on peut facilement dégager un axe principal. Cette approche permet de définir facilement le long d'un axe paramétré, par le biais de transformations affines dans le repère courant local, des surfaces relativement sophistiquées. Dans le cas de la simulation d'objet filaires, cette modélisation permet trés facilement d'ajouter une épaisseur, par extrusion de section. Le problème principal de ces modèles, à propos duquel aucune solution ne fait pour le moment l'unanimité, est la question de la gestion des points de fortes courbures sur l'axe: ceci peut, si la tesselation n'est pas adaptée, générer des maillages de mauvaises qualités (cf Figure 1.1), voire même des maillages auto-intersectant (CF. figure 1.2).

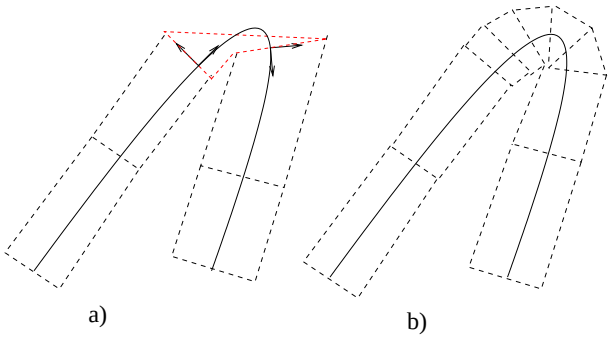


Figure 1.1: a) tesselation incorrecte du cylindre généralisé; b) tesselation correcte.

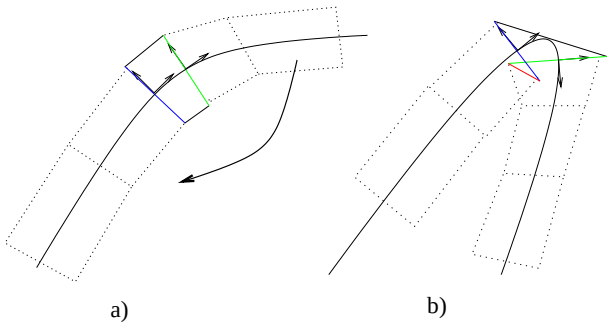


Figure 1.2: Le cylindre maillé en a), dans le cadre de déformation, génère un maillage auto-intersectant en b).

1.1.2 objets surfaciques

La simulation physique utilise ces objets pour afficher des surfaces ouvertes (ex: tissus), mais aussi fermées, représentant des volumes: il est en effet courant, dans le cas de simulations ne manipulant pas de changements topologiques de type découpe ou déchirure, d'utiliser une surface comme "peau" d'affichage d'un volume animé. Les différents modèles classiques que l'on trouve pour la manipulation géométrique de surface sont:

- Les maillages polygonaux: ces modèles sont les plus utilisés à l'heure actuelle, de par leur simplicité, et le fait qu'ils collent de trés prés aux modélisation mécaniques surfaciques courantes (voir section 1.3). Leur manipulation, en tant que modèle géométrique, s'énonce assez simplement, et les cartes graphiques actuelles en font leur outil d'affichage privilégié. Suivant les tâches algorithmiques à effectuer, la structure de données utilisée est toutefois une question sensible: d'une manière générale, la plupart des manipulations topologiques requierent, pour être accomplies dans de bonnes conditions, de disposer efficacement des informations de voisinage. On doit souvent, si l'on veut éviter une structure de donnée trop redondante, circonscrire les modifications qui seront effectuées [BBCK03]. Notamment, la question de savoir comment séparer un objet en deux parties non-connexes, et manipuler une structure de donnée qui permette de détecter ce découpage de manière efficace pour le contexte de la simulation temps réel, est une question encore à notre connaissance ouverte.

- Les splines: les surfaces splines sont le plus souvent construites par produit tensoriel [Gri99]. Ces objets sont de fait paramétrés; ils bénéficient de plus, comme les splines 1D, de certaines propriétés mathématiques, comme par exemple la propriété d'insertion de point, qui rendent le modèle attractif pour les aspects adaptif/multirésolution (cf section 2.2.1). Toutefois, il est à noter que l'aspect tensoriel de la construction induit un certain nombre de difficultés pratiques: cette construction implique de fortes contraintes topologique, rendant par exemple impossible tout autre découpe que complète suivant une direction paramétrique principale[Gri99]. Il est de plus à noter que l'assimilation des lignes brisées à un cas particulier de spline (ce qui était le cas pour la description des modèles 1D) n'a pas été faite, dans le cas des surfaces: la topologie d'un maillage, dans la littérature, est le plus souvent arbitraire, et nous considérerions comme abusif d'assimiler de tels objets à des cas particuliers de splines comme nous les envisageons ici, i.e. définie, pour les surfaces, par produit tensoriel. Il est à noter que certaines modèles spline sont construits pour fonctionner sur un maillage triangulaire [Man03], et d'autres, telles les T-splines [SZBN03] définies récemment, proposent une technique de construction de "patch" spline associée à des méthodes efficaces de recollement, qui permettent de manipuler des objets de topologie arbitraire. Ces techniques sont toutefois encore trés récente, et l'utilisation de splines surfaciques sans l'utilisation de construction tensorielle est, à notre perception, encore isolée.

- surfaces à subdivision: elles sont construites de manière itératives, d'une manière trés semblable aux courbes à subdivision, en adaptant cette fois le schéma de subdivision à des objets polygonaux. La paramétrisation de ces surfaces est disponible via les mêmes outils que pour les courbes à subdivision, moyennant les même restrictions sur le schéma de subdivision. Dans le cas des surfaces, la topologie du maillage de départ est, là aussi, un élément limitant: la structure de la matrice de subdivision est dépendante de la valence des points du maillage initiale. La structure spectrale en est, de fait, elle aussi dépendante.

Ceci impose, meme si la théorie ne le fait pas, une certaine régularité sur la topologie du maillage, pour pouvoir être paramétrée (relativement) simplement.

- surfaces définies par nuage de points: l'idée fédératrice de ces techniques est que, sous couvert de disposer d'un échantillonnage suffisamment dense d'une surface, on peut la reconstruire. Ainsi, des techniques de reconstructions locales (telles les *Moving Least Square* [MACOF+03]) ou globales (fonctions de Green[VSK95], RBF[CBC+01]) permettent d'approximer la surface. Ces techniques, quand elles envisagent l'objet dans sa globalité, sont généralement assez couteuses (dans le cas des RBF, le système à résoudre est par exemple assez mal conditionné, rendant à priori délicat l'utilisation d'une eventuelle cohérence temporelle, dans le cadre d'une reconstruction interactive). Pour finir, la topologie résultante de l'objet reconstruit est assez délicate à contrôler finement (voir section 2.2.2 pour plus de détails).

1.1.3 objets volumiques

en terme d'affichage, la frontière entre modèles surfaciques et volumiques est assez délicate à préciser. Nous considérons que la modélisation géométrique devient volumique dans deux cas: lorsqu'on associe à un objet la possibilité d'être découpé ou fracturé, et ensuite, lorsque la modélisation est celle d'un champ de densité (ex: gaz). Dans ce dernier cas, il est important de garder en tête que ce qui est décrit ici est uniquement orienté vers l'affichage: on pourra par exemple retrouver les maillages 3D comme discrétisation de la mécaniques (pour éléments finis ou autre) mais ceci sera traité dans la section 1.3

- représentations implicites : les représentations implicites sont, comme décrit pour le cas 1D, un moyen générique d'associer un volume à un squelette de dimension quelconque. Ceci est trés utile pour les objets volumiques déformables[FLA+05] (voire de simulation d'objets trés déformables [DC95]), mais aussi pour la reconstruction géométrique de certains phénomènes type surface de fluides. La gestion topologique simple de ces représentations est, dans le cas de la simulation de fluide, clairement un avantage, dans la mesure où elle permet notamment de gérer de manière systématique le caractère partiellement continu du milieu (on peut par exemple voir [GSLF05] pour des simulations, non interactives, de fluide ou le caractère partiellement continu du milieu est mis en évidence).
- représentations par maillages 3D : la plus courante est la représentation par tétrahèdres. Celles-ci peuvent indifférement jouer le rôle de cellules d'echantillonage pour un champ de densité (pour les modélisations de type gaz [AN] en modélisation lagrangienne par exemple, voir section 1.3), où d'éléments de volumes pour des objets déformables. Dans ce cas, on peut alors par exemple associer l'objet à une opération de découpe [For03] ou de fracture[MTG04].
- représentations par points: ce type de représentation est utile pour la simulation des gaz (cette fois dans une approche eulérienne, cf section 1.3): on obtient alors un système particulaire, qui peut pour l'affichage, être mis en correspondance avec un champ de densité, et affiché par rendu volumique [Sta99].

La géométrie utilisée pour l'affichage dans le cadre des simulateurs physique pose, de manière générale, des problèmes qui ne sont pas ceux de la modélisation géométrique dans sa version la

plus classique. La question essentielle, dans le cadre de la simulation physique, est surtout de garder des procédures d'affichages rapides. Les cartes 3D font que cette tâche n'est clairement pas le bloc fonctionnel limitant: on se limite le plus souvent à des configurations géométriques simples. Toutefois, certains cas de figure donnent tout de même des problèmes intéressants (voir la contribution sur l'affichage temps réel des cylindres généralisés, section 2.5.1), parce que l'on souhaite idéalement obtenir un affichage qui puisse autant que possible pallier aux approximations souvent nécessaire sur le modèle mécanique, pour conserver un objet qui puisse être manipulé en temps interactif. Dans cette optique, on peut penser que la conception de simulateurs physiques, assez paradoxalement, pourra durablement alimenter une activité de recherche sur la géometrie de manière perenne.

1.2 Modèle de collision

la question de savoir comment modéliser l'interaction entre objets est une question cruciale, car souvent très coûteuse, en simulation physique. A la différence des autres classes de modèles envisagées dans ce document, un modèle de collision est partagé par un groupe d'objet, et n'appartient de fait pas "strictement" à un système mécanique donné. Cette particularité rend difficile le fait d'espérer un jour obtenir un modèle de collision universel (nous reviendrons sur ce point dans la section 3.1.3. Comme nous le verrons, il est difficile de dissocier la détection de collision proprement dite, de la réponse à celle-ci, i.e. de la manière dont cette collision influe sur les équations mécaniques. C'est pour cette raison que nous traitons les deux points au sein d'une même partie.

1.2.1 Detection de collision

on distingue deux grandes classes de techniques [Boy79]: d'une part, les techniques qui considére le mouvement comme une suite discrète de dates, en cherchant ensuite les "collisions" (i.e. intersections) à chaque date (elles sont classiquement appellées *méthodes de collisions à temps discret*. D'autre part, les techniques qui vont tenir compte de la continuité du mouvement, en cherchant, sur la base de la ligne de temps discrète, les dates de simulation où les objets entrent en contact. Ces méthodes sont dites *à temps continu*. Ces dernières techniques sont souvent plus coûteuses, mais plus précises [Red04]. Nous tentons ici un rapide survol des techniques modernes de détection de collision[JTT01, TKH$^+$05, Mes02]. On peut citer:

- primitives sphériques: il s'agit trés probablement de la primitive de collision la plus simple, le test d'intersection de ramenant ici a une simple distance entre deux points. Ces primitives sont adaptés à des modèles type matériaux granulaire [BYM05], mais, dans la mesure où un ensemble peut de sphère peut approximer de manière arbitraire n'importe quel objet, ces primitives peuvent aussi être utilisée pour un ensemble plus divers d'objets [Mes02].
- détection entre polyèdres: les objets sont ici considérés comme un ensemble de polyèdres, ce qui ramène la question de l'intersection entre les objets, à un ensemble de paires potentielles de polyèdres. La plupart du temps on se ramène à des polyèdres convexes. L'algorithme GJK [EGK88, Cam97], basé sur l'utilisation de la distance de Minkowski entre deux polyèdres, évalue de manière itérative la distance d'interpénétration. Cet algorithme est de complexité linéaire par rapport au nombre de sommets des deux polyèdres,

mais dans certains cas il génère certaines instabilités (ex: un cube quasiment au repos sur un plan). Dans le cas où l'on cherche une évaluation du volume d'intersection, l'utilisation de cette primitive peut être, pour obtenir un traitement rigoureux de l'intersection, assez coûteuse et nécessiter pas mal de cas particulier [Mes02]. Dans la mesure ou ces primitives peuvent servir de base de définition à une mécanique déformable, ils peuvent, dans l'absolu, servir de primitive de collision pour les objets déformable, mais leur utilisation pratique reste, à notre connaissance, assez coûteuse, et délicate pour le temps réel.

- utilisation des cartes de distances: ces algorithmes se basent sur une approximation discrète de la fonction qui, en tout point de l'espace, donne la distance entre le point et l'objet considéré[MAC04]. A partir de cette carte discrète, on dispose d'un moyen simple pour connaitre le positionnement intérieur/extérieur d'un point donné de l'autre objet, et on peut aussi avoir une estimation de la distance d'inter-pénétration. Ces cartes, trés efficaces dans le cas d'objets rigides, sont pour le moment encore relativement difficiles à utiliser dans le cas d'objets déformables.

- utilisation de représentations implicites: ce type d'approche est assez proche, philosophiquement, de la technique précédente. Une représentation implicite fourni un moyen, souvent peu coûteux, pour évaluer l'aspect booléen intérieur/extérieur d'un point de l'espace par rapport à l'objet considéré. Par recherche d'extréma, cette approche permet aussi de déterminer la distance entre deux objets, ou la distance d'interpénétration si les deux objets sont en intersection [SP95]. Pour évaluer le volume d'intersection (si nécessaire, voir section 1.2.2), on peut, par une tesselation locale, évaluer ce volume [DC94], pour en déduire une réponse ad-hoc.

- collision entre volumes extrudés: ce type de problématique survient essentiellement quand on veut, pour la détection de collision, prendre en compte l'aspect continu du mouvement. Chaque primitive de collision intervient donc non pas directement, mais par le biais d'une extrusion, et l'on se ramène par ce biais à la détermination de l'intersection de deux primitives géométriques: au travers de la détermination d'une intersection, une question majeure est de trouver le point d'extrusion, i.e. la date de simulation, à laquelle les objets entrent en collision [RKC01, RKC02a].

- méthodes non-géometriques: une famille de méthode a récemment été proposée, tirant parti des architectures matérielles 3D récentes [GRLM03, Man05]. Ces techniques ne manipulent pas à proprement parler de primitives géométriques, mais utilisent d'une part l'extrème optimisation des cartes graphiques pour discrétiser la géometrie, et d'autre part la capacité des cartes les plus récentes pour signaler qu'une partie de la géometrie est tracée *derriere* la caméra openGL. De la sorte, le positionnement de la caméra sur un triangle d'un objet A, orienté dans la direction de la normale à ce triangle, peut fournir simplement une information d'intersection avec un objet B: simplement en effectuant un rendu de l'objet B. En associant de plus à chaque primitive (ou groupe de primitive) une coloration spécifique, différente de ses voisins [BEH02], on peut également par ce biais détecter efficacement les auto-collisions dans le cas d'objets déformables.

1.2.2 La réponse à la collision

La réponse à la collision est la partie algorithmique qui, en fonction de l'examen géométrique décrit à la section précédente, permet d'en déduire une "correction", par le biais des lois

mécaniques, de l'état de l'objet.

- méthodes à pénalité: il s'agit de la classe de méthode la plus simple à mettre en place. Elle s'appuie la plupart du temps sur le vecteur d'inter-pénétration (le vecteur de translation minimale permettant de séparer les deux objets) entre deux objets, et génère une force de répulsion calculée en fonction de ce vecteur (de manière simple, linéaire[MW88] ou non-linéaire, simulant des forces autorisant moins d'interpénétration).

- méthodes à contact exact: ces méthodes tentent de fournir une simulation qui, au long des différents temps de simulation, donne à chaque date une configuration géométrique valide (i.e. sans interpénétration). Ceci, dans le cas de point de contact unique entre objets rigides, peut se faire selon des règles relativement bien maîtrisées [Hah88, MW88]. Toutefois, dans le cas de contact multiples, le problème devient plus complexe, et l'on se ramène souvent à une résolution itérative [Bar94, RKC02b]. Dans le cas des objets déformables, une correction géométrique est appliquée aux objets [Gas93] et la cinématique locale des objets adaptées en fonction du type de contact[PPG04].

- gestion des frottements: dans le cas d'objets en contact, il est possible, pour certains cas simples, d'utiliser les lois de Coulomb pour modéliser le frottement. Ceci donne de bon résultats dans le cas de contact ponctuel. Dans le cas où il y a plus d'un point de contact, ou une zone continue de contact, la résolution de ces frottements devient un problème NP-complet [Bar91]: la solution envisagée alors est souvent de gérer le contact par une approche autorisant de (trés faibles) interpénétrations, ce qui permet de gérer des contacts avec frottements bien plus complexes [Dur04a, KEP05].

- gestion de contacts entre plusieurs objets: ce cas est assez particulier, dans la mesure ou les techniques décrites dans les points précédents est relative à la gestion du contact entre deux objets: dans le cas de contact entre plusieurs objets, le système est soumis a un nombre plus élevé de contrainte, et pour trouver une réponse satisfaisante, le système mécanique contraint doit alors être envisagé dans sa globalité. Certaines approches [Fau96] résolvent le problème par itération, pour tenter de converger vers une solution physiquement plausible.

La détection de collision est un domaine extrèmement actif en recherche, et aucune solution ne fait pour le moment l'unanimité de la communauté. L'essentiel de la recherche dans ce domaine vise à proposer des modèles qui soient aussi génériques (et aussi rapides) que possible. Il semble pour le moment inaccessible d'obtenir un jour un modèle "universel", qui soit capable de gérer de manière rigoureuse un nombre suffisant de types d'objets pour que la question de la collision soit considérée comme résolue. En revanche, les modèles de collisions de la littérature sont souvent assez complémentaires (ex: primitives sphériques rapides mais grossières, primitives polyédrales précises mais coûteuses; détection discrète rapide mais dépendante de la discrétisation du temps, détection continue fiable, mais plus complexe à mettre en place). Cette constatation laisse à penser qu'au delà d'un hypothétique modèle de collision universel, un système mécanique pourrait d'ores et déjà bénéficier d'un traitement à la fois souple, peu coûteux, et robuste, s'il disposait non plus d'un seul mode de gestion de la collision, mais par la cohabitation de plusieurs algorithmiques. Cette idée rentre directement dans l'optique visée par la notion de multi-modèle, nous y reviendrons section 3.1.3.

1.3 Modèle mécanique

Cette section regroupe la partie d'algorithmique nécessaire à la simulation qui, à partir du bilan de force s'appliquant sur un objet, permet d'en déduire les lois de modifications (le plus souvent, une équation différentielle) des variables décrivant le système: ces variables sont ce que l'on appelle classiquement les *variables d'états*. Que l'on soit en temps discret ou en temps continu, la tâche est ici, étant donnée les variables d'états à une date de simulation donnée t_i, d'arriver à évaluer les variables d'états à la date de simulation suivante $t_i + \delta t_i$ (δt_i pouvant être, suivant la sophistiquation de la simulation, constant ou variable, voir). les techniques d'intégration numériques des équations différentielles contraignent souvent, pour être stables, δt_i à être de l'ordre de quelques centièmes de secondes, dans le meilleur des cas, pour la plupart des simulations adressées en simulation chirurgicale. Il est à noter que d'une manière générale, il est bien souvent, en tenant compte des technologies actuelles (autant matérielles qu'algorithmiques), impossible d'obtenir un temps de calcul pour une boucle de simulation (i.e. phase de collision + phase mécanique) qui soit inférieur ou égal au pas de temps δt_i. C'est pour cette raison que l'on appelle généralement *simulation temps réel* une simulation garantissant un niveau d'interaction suffisant, généralement une trentaine de boucle de simulation par secondes (hors retour d'effort). Ce taux de rafraîchissement est pour le moment, hormis des cas de simulation trés simples, impossible à garantir, et l'on est souvent limité par le temps de calcul de la phase de détection de collision, et la stabilité des méthodes d'intégration numériques (induisant alors pour être stables une algorithmique coûteuse): bien souvent, le taux de rafraîchissement doit être revu à la baisse. Il en résulte un décalage entre le temps de simulation *perçu* par l'usager, et le temps de simulation effectivement utilisé: ceci induit une "dilatation" du temps entre l'homme et la machine. Cette dilatation a le premier inconvénient de fausser la perception d'un modèle mécanique (même rigoureux d'un point de vue théorique), dans la mesure ou les mouvements calculés sont ralentis, vu par l'humain. De manière duale, vu de la simulation mécanique, cette dilatation du temps induit une augmentation de l'influence des interactions de l'utilisateur (ex: un interacteur bougé à une certain vitesse par l'utilisateur aura une vitesse amplifiée pour la simulation), faussant ainsi les forces appliquées, et rendant souvent incertaine la stabilité des méthodes d'intégration numériques.

Si l'on se base sur la bibliographie, il semble difficile, à l'heure actuelle, de dissocier, au sein de la "mécanique", trois aspects: d'une part la question de savoir comment, à partir des forces appliquées sur un système, en déduire l'equation différentielle qui va régir le mouvement (ce que nous appellerons ici *lois mécanique*), la manière dont on va utiliser ces équations pour déterminer le mouvement (l'intégration numérique), et comment on va pouvoir pallier aux limites des équations différentielles, pour simuler des contraintes fortes entre objets. Ces trois aspects sont en effet intimement liés (ex: le cout d'une méthode d'intégration numérique, et sa stabilité, sont liés au type de système qu'il intègre, i.e. à la loi mécanique utilisée). C'est pourquoi nous avons choisi de les traiter au sein d'une même partie *mécanique*.

1.3.1 lois mécaniques

Les différents modèles mécaniques que l'on trouve dans la littérature peuvent être catégorisés de manière naturelle par le biais du degré de déformabilité du corps modélisé [Hab97]:

- rigide: On considère souvent que la mécanique des corps rigide est globalement mieux

résolue que celle des autres classes de modèles de cette section [GBF03]. C'est pour cette raison qu'au delà de la mécanique classique, on peut trouver pour ces modèles des travaux tentant certains apports non-purement physiques, tels une parallélisation de la simulation, avec gestion rigoureuse des collisions[Mir00].

- solides déformables: on considère ici les objets déformable ayant toutefois une certaine rigidité. Ces objets ont la particularité d'être, s'ils sont considérés comme des objets déformables classiques, soumis à des termes energétiques importants: ceci résulte souvent en une intégration numérique délicate. La théorie des poutres [Cou80] défini de manière mécaniquement rigoureuse le comportement infinitésimal d'un modèle dont l'une des dimensions est largement supérieure aux autres (rendant ainsi l'objet assimilable à un objet 1D). Il est à noter que c'est par le biais de la section, donc d'aspect intrinsèquement volumiques, que les propriétés mécaniques d'une poutre sont calculées. Les frères Cosserat [EC09] ont complété le modèle, en associant aux éléments de torsion et flexion (déjà présent dans la théorie des poutres) des paramètres de cisaillement et d'élongation. Cette théorie a récemment été reprise, dans un cadre simplifié, pour modéliser en temps réel un fil relativement rigide[Pai02].

- déformables structurés : ce cadre est celui que l'on adresse le plus souvent lorsqu'on parle de simulation physique d'objets déformables. La littérature est ici extrêmement large, mais l'on peut dégager les familles de techniques suivantes:

  - systèmes masses-ressort: l'objet est ici discrétisé en un ensemble de particules, lesquelles particules sont reliées (pour certaines) par un ressort, i.e. une liaison imprimant une force de rappel, fonction de l'élongation par rapport à une longueur au repos. Les forces internes au modèles sont ainsi approximées, sous couvert d'un positionnement correct des ressorts. Dans la plupart des cas, les ressorts utilisés sont *linéaires*. Il est à noter que l'on peut utiliser des liaisons similaires pour mettre en place des forces de rappel en courbure, et/ou en torsion. De tels systèmes mécaniques permettent de mettre en place des modèles mécaniques déformables relativement sophistiqués [Pro95]
  - éléments finis: ces méthodes utilisent des primitives volumiques, et modélisent les déformations de ces structures élémentaires, en fonction des efforts qui leur sont appliquées[DCA99, PDA00]. Ces méthodes sont le plus souvent non-invariante en rotation, a contrario des méthodes masses-ressort: de fait, pour gérer de grands deplacement, des méthodes à base d'extraction corotationnelles doivent être employées [THMG04]. Ces méthodes s'adapte bien aux opérations de découpe [For03].
  - analyse modale: cette classe d'approche pré-suppose que les déformations possible pour un objet sont limitées, et précalculables[JP02]. Il est à noter que, dans le cas d'éléments finis linéaires, un certain nombre de précalcul sont possibles, rendant ainsi la technique assez proche de la classe d'analyse modale[Cot97], quand bien même ces dernières techniques ne sont classiquement pas mises dans cette classe de techniques.
  - meshless: ces techniques sont apparues il y a peu en informatique graphique [MKN+04]. Elles utilisent, pour représenter mécaniquement un objet, une nuage de point, sur lequel aucune structure n'est pré-imposée: seul un calcul basé sur les distances de voisinage permet d'effectuer le bilan de force sur chaque élément. Il est à noter que, si ces techniques peuvent potentiellement gérer des opérations de découpes [PKA+05], on ne sait pas, à notre connaissance, effectuer de telles manipulation de manière

précise en temps réel (le principal frein étant l'intime liaison de ces techniques avec la reconstruction géométrique de la surface visualisée, délicate à contrôler topologiquement dans le cas d'une découpe).

Il est à priori délicat de parler ici de méthodes *continues* et de méthodes *discrètes*: toutes les méthodes décrites se ramènent effectivement à un ensemble discret, fini, de degrés de libertés. Toutefois, Il est, pour certaines méthodes parmi celles citées, plus facile d'appliquer un effort en un point quelconque de la structure (et non nécessairement directement sur les degrés de libertés. A partir de ce critère (arbitraire), on pourra parfois, dans ces techniques, séparer les techniques de mécanique dites *continues* (éléments finis, meshless, splines dynamiques) des méthodes *discrètes* (masses-ressort, analyse modale).

- corps non-structurés: cette classe de modèles englobe les modèles tels les fluides, fumées, tas de sables.
  - approche lagrangienne: pour ces méthodes, des particules sont utilisées pour discrétiser l'objet, et l'animation de ces particules génère, par reconstruction, celle de l'objet. Ceci est particulièrement adapté pour l'animation de tas de sables [BYM05], ou l'interaction fluide-solide [Fed02]. Il est à noter que cette classe d'approche englobe également les modèles structurés: on utilise toutefois assez peu la terminologie, dans la mesure ou l'approche eulérienne, décrite ci-dessous, n'a, elle, pas de transposition simple dans le cas d'objets déformables structurés.
  - approche eulérienne: cette famille d'approche utilise un partitionnement de l'espace, en chaque point duquel on défini l'évolution, au cours du temps, des paramètres cinématiques qui définissent le mouvement. Le contexte d'application le plus propice est celui de l'évolution d'un flux dans un espace clos [Sta99].

1.3.2 modèles de résolution des équations mécaniques

A partir de l'équation différentielle définissant le mouvement, les modèles décrits ici vont proposer des solutions calculer les nouvelles valeurs du vecteur d'état. La question de la résolution numérique d'équation différentielle est une question trés abondamment traitée en mathématiques appliquées: toutefois, une spécificité du domaine réduit le nombre de techniques utilisables: le bilan de force appliqué sur un système évolue en fonction du temps, et de fait l'équation différentielle à résoudre n'est jamais complétement connue. Cette spécificité rend par exemple inaccessible pour le moment les (nombreux) résultats de résolution hiérarchique d'équations différentielle [Coh00], utilisant entre autres les travaux de Galerkin sur l'utilisation de bases d'ondelettes pour la décomposition d'opérateurs linéaires[Coh00]. On est de fait la plupart du temps ramené à des techniques classiques pour cette classe de problèmes.

- résolution statique: ces classes de méthode ne résolvent pas à proprement parler l'équation différentielle obtenue, mais recherchent uniquement un *état d'équilibre* [BNC96, NS00]: les vitesses et accélérations des degrés de liberté du système sont donc ignorés. Sous couvert d'approximation linéaire, on se ramène de fait à un système simple à résoudre. Dans le cas de mouvement simples, pour lesquels les états transitoires n'ont pas d'importance, ces méthodes peuvent donner de bon résultats: en assimilant chaque date à un état d'équilibre, on peut ainsi simuler un mouvement (approche *quasi-statique*). Toutefois, les phénomènes

type propagation d'onde de déformation, chute de corps, phénomènes pendulaires, etc... sont totalement inaccessibles par ce type de méthodes, ce qui peut être un inconvénient important suivant le type de phénomène que l'on cherche à reproduire. Certains travaux tentent de pallier à cet inconvénient [D'A01] en utilisant un processus de résolution itératif de l'état d'équilibre, et en utilisant (de manière approximative) les résultats des itérations comme état transitoire. Ces techniques peuvent être considérée comme assez grossières, moins rigoureuse physiquement, en comparaison des techniques dynamiques décrites ci-dessous. Elles sont toutefois souvent plus simples à gérer, et donnent, dans le cas d'objets fixes déformables, de bon résultats.

- résolution dynamique: ces méthodes résolvent de manière complète l'equation différentielle régissant le mouvement du système mécanique considéré.
  - méthodes explicites: ce sont les méthodes les plus simples, les plus célèbres étant la méthode d'Euler explicite, et les méthodes de Runge-Kutta. Ces méthodes, peu coûteuses, souffrent souvent d'une faible stabilité numérique. Cette instabilité, dans le cadre de la simulation temps réel, cause des problèmes sévères si elle n'est pas maitrisée [Pro95, Jou95] (provoquant typiquement un comportement irréaliste, le plus souvent de trés forte cinématique et complètement incontrôlable, des systèmes mécaniques). La stabilité des techniques explicite est dépendante du pas de temps utilisé, et l'on est souvent limité à des pas de temps assez faibles.
  - méthodes implicites: ces méthodes ont en commun de chercher le nouveau vecteur d'état en manipulant le bilan de force *à la prochaine date du système*: dans la mesure où ces forces dépendent clairement de la configuration des variables d'états, ces dernières deviennent solution d'un système d'équation (non linéaire). Toutefois, le résultat obtenu est clairement meilleur que pour les techniques implicites dans la mesure ou le vecteur d'état est calculé à partir des forces qu'il va générer dans sa nouvelle version. Ces techniques sont plus stables que les techniques explicites, et autorisent des pas de temps plus larges, ce qui les rend particulièrement attractives en simulation temps réel pour les objets déformables complexes (ex: tissus) [BW98, BWK03].

1.3.3 Contraintes

un bilan de force est la seule dont on devrait, pour mener à bien une simulation, avoir besoin, dans la mesure ou la relation entre système mécanique s'exprime, dans la réalité, uniquement par ce biais. Toutefois, dans certains cas, les relations mécaniques se présentent plus comme des *conditions* externe (ex: point d'un objet contraint à rester en un point donné de l'espace, relation de pivot, point pouvant glisser le long d'une barre, etc...): ces situations ont ceci de particulier que le bilan de force n'est pas ce qui est le plus facile à évaluer. Dans ces configurations: il est souvent aisé de spécifier, avant même l'intégration, une partie des degrés de libertés: on connait déjà, avant intégration, une partie du résultats que l'on souhaite obtenir pour la configuration. La gestion de ces *contraintes* est donc souvent gérée à part, par des techniques différentes de celles liées à la collision, et au calcul de réponse:

- projection: ces méthodes tentent, aprés la phase d'intégration numérique, de corriger (de manière directe, ou bien encore par exemple par moindre carré) les positions, vitesse et

accélération pour garantir le respect des contraintes. Ces techniques donnent de bon résultats dans le cas de contraintes simples. Toutefois, les corrections ont quelques inconvénients: d'une part, les corrections doivent être faites de manière à ne pas rendre le système d'intégration instable à l'étape suivante. D'autre part, dans le cas de certains contraintes sophistiquées (ex: cycles), la question des corrections est une question délicate à résoudre. Cette classe de méthode est la plus générale parmi celles mentionnées ici: les autres techniques mentionnées (pénalité et multiplicateurs de Lagrange) peuvent être vus comme des cas particulier, utilisant uniquement des termes correctifs en force.

- méthodes à pénalité: il s'agit de la méthode de gestion la plus simple. Elle revient à utiliser, pour chaque contrainte, un ressort [MW88], tentant de manière lâche de faire respecter la contrainte. Ces techniques sont faciles à mettre en place, mais créent des contraintes "molles": les raideurs des ressorts doivent souvent rester faible, sous peine d'introduire des termes energétiques trop importants, et de rendre la phase d'intégration numérique délicate.

- multiplicateur de Lagrange: ces méthodes déterminent les forces à insérer dans le système en les gérant comme des inconnues, ajoutées aux degrés de libertés du système. Ces nouvelles inconnues sont ainsi intégrée en même temps que l'equation du mouvement. La contrainte, pour être intégrée de manière souple (i.e. pour optimiser la convergence, en ménageant la stabilité du système) est le plus souvent intégré au système dynamique par le biais du schèma de Baumgarte [Bau72], reliant entre autre l'expression de la contrainte au pas de temps utilisé pour l'intégration numérique. Ces techniques envisagent un système de manière globale, et les structures de cycles sont par exemples correctement gérées [LF04]. Il est à noter que ces techniques peuvent être vues comme un calcul particulier de corrections [Fau98] et que, de fait, ces techniques peuvent être vues comme un cas particulier des techniques de projection citées plus haut.

La mécanique est, avec la détection de collision, le domaine de publication privilégié de la simulation physique temps réel. L'essentiel de la problématique est, ici, de fournir au temps réel des modèles mécaniques aussi rigoureux et stables que possibles. Cette thématique, très riche, comporte encore de bien beaux jours devant elle, car on est encore très loin d'une simulation physique collant (par exemple pour les fluides ou les corps déformables) à la perception que nous avons du réel.

1.4 Modèle d'interaction

l'interaction est un aspect de la simulation physique pour le moment encore émergeant (si l'on examine le ratio d'études sur la question, dans le volume de publication sur la simulation physique temps réel). Par modèles d'interaction, on peut désigner ici l'interface homme-machine, que certains travaux adressent encore embryonnairement [MLFC04]. L'essentiel des résultats produits ici regroupent, à notre perception la question du contrôle des périphériques à retour d'effort [OL05, Dur04b]. La difficulté principale pour ce contrôle, dans le cadre de la simulation physique, est le besoin de performance importante: pour être correctement contrôlé, on considère généralement que la fréquence de mise à jour du périphérique doit être supérieure à 300 Hz, ce qui est bien au delà de ce que peut pour le moment générer la simulation physique temps réel. Pour ce faire, on utilise souvent la technique suivante: à coté de la simulation, un thread autonome contrôle le système haptique, découplant ainsi la fréquence de contrôle de celle

de simulation: un système d'interpolation peut alors être utilisé pour envoyer au périphérique les informations. Ceci introduit un (léger) décalage temporel entre la simulation et le contrôle haptique, mais garanti un contrôle stable et haute fréquence du périphérique.

les modèles d'interaction, s'ils ont été cité dans ce chapitre, sont l'objet d'une étude encore émergeante dans le domaine de la simulation physique. Ce point est clairement multidisciplinaire, et la question de la mise en correspondance du virtuel avec les sens humains réel est une question difficile, encore peu traitée. Nous ne reprendrons pas, dans les prochains chapitres, cette partie des modèles nécessaires à la simulation, dans la mesure où d'une part notre activité ne s'y est pas pour le moment intéressé, et d'autre part parce que nous ne connaissons pas, dans la bibliographie existante, de travaux pouvant être percus comme une quelconque adjonction d'adaptivité, à forciori de multi-modèle, à ces techniques.

1.5 Contributions

1.5.1 Une technique rapide de contrôle de mélange pour représentation implicite [TGMC03]

Comme mentionné dans la section 1.1.1, les représentations implicites sont un outil géométrique permettant de générer, par le biais de squelettes une surface associée à un certain volume. La propriété de mélange est, dans le contexte de l'habillage de modèle 1D, un avantage clair de la représentation implicite, dans la mesure où elle garantit, quelque soit la déformation du squelette sous-jacent, une continuité élevée de la surface reconstruite. Pour pouvoir contrôler cette propriété, l'approche standard est d'utiliser un graphe de mélange, pour lequel les sommets sont les primitives, et les aretes les relations de mélange autorisées [GW95]. La gestion de groupes de mélange classique pose toutefois quelques problèmes: d'une part, elle ne gère pas les cas de relation de mélange *partielles*, où la propriété de mélange de deux groupes dépend de la localisation (ex: pour la reconstruction d'un humanoïde, le bras et le cors doivent se mélanger au niveau de l'epaule, mais pas au niveau de la hanche, si le bras est placé le long du corps). D'autre part, une des techniques réputées les plus rapides d'affichage d'objets implicite, le marching cube [LC87], ne peux pas, dans sa formulation classique, tenir compte de primitives ne se mélangeant pas (il est d'ailleurs une observation classique que de constater que le marching cube ne garanti pas, dans sa version de base, une adéquation topologique fine avec l'isosurface à reconstruire). [TGMC03] propose une solution à ces deux problèmes, en ramenant la manipulation du graphe de mélange au cube de tesselation. Ceci résoud d'une part le premier problème (de gestion des relations de mélange partielles), et permet une adaptation de l'algorithme de marching cube, proposé dans la deuxième partie de l'article. Cet article propose également une astuce de codage qui peut être, dans certains autres cas, intéressante: le codage classiquement utilisé pour coder un graphe de mélange (de manière générale n'importe quel graphe non-valué) est une matrice (creuse) contenant uniquement des valeurs booléennes. La technique décrite dans [TGMC03] utilise des mots binaires pour stocker cette matrice, et ramène les opération nécessaire sur cette matrice (existence d'une relation de mélange; insertion; suppression) à des opérations logiques bit-à-bit sur ces mots binaires, accélérant ainsi de manière significative le traitement de ces opérations au niveau du processeur. Nous pensons que cette technique pourrait être avantageusement utilisée dans d'autre contextes que celui présenté. Il est par exemple probable qu'une telle technique puisse permettre d'étendre les techniques de détection de collision stochastiques [RGF+04] pour permettre un marquage des primitives déjà traitées, et garantir

une gestion rigoureuse et complète des collisions en n pas de temps.

1.5.2 Un modèle de fil B-spline déformable en temps réel[LGM+04]

L'activité portant sur la simulation temps réel de fil chirurgical est une activité initiée par le travail de thèse de J. Lenoir. Nous avons choisi, parmi les publications déjà générées par ce travail, d'inclure dans ce rapport d'habilitation la publication portant sur les contraintes glissantes, dans la mesure où nous considérons que celle-ci donne (même si pour certains points, notamment la définition des énergies, de manière trés compacte) un bon aperçu de ce qui a été produit jusqu'à présent sur ce modèle. Ce modèle utilise comme support de la mécanique les *splines dynamiques*, défini par Rémion [RNG99]. Leur avantage principal, par rapport au classique système masse-ressort, et de pouvoir (ce qui est une propriété classique des splines) représenter avec un nombre faible de degrés de liberté des configurations géométriques sophistiquées: les fonctions de bases associées aux points de contôle étant de degré polynomial supérieur à celles utilisées par les systèmes masses-ressort, l'approximation générée est meilleure. Qui plus est, ce modèle intègre la possibilité, étant un point d'application de force quelconque, de répartir l'effort exercé sur les points de contrôle environnant: Cette fonctionnalité fait que ce modèle est un modèle mécanique continu. A ce modèle ont été associés des termes d'évaluation d'énergie cinétique interne (élongation et flexion) qui permettent de simuler en temps réel un objet déformable 1D. Un jeu de contraintes, résolues par multiplicateur de Lagrange, a été proposé: contraintes d'un point de la courbe sur un plan (éventuellement avec gestion de friction [7]), un segment de droite, contraintes en tangentes et courbures, et pour finir, contraintes de point glissant: ces contraintes sont cruciales pour espérer modéliser l'opération de suture. Cette contrainte de point glissant présente la particularité, contrairement aux autres contraintes présentées, plus classiques, d'imposer une contrainte sur un point dont l'abscisse paramétrique n'est pas spécifiée, et qui fait donc partie des inconnues. Cette contrainte permet de simuler le passage d'un fil au travers d'un point d'une surface donnée, et rend de fait potentiellement accessible la simulation d'une suture. Nous pensons que la mécanique spline continue est un candidat particulièrement attractif, dans la mesure où le niveau de continuité, et donc la souplesse apparente, du fil, est d'un niveau suffisamment élevé pour un nombre de degré de liberté réduit, et donc permet des temps de calcul raisonnable combiné à une représentation lisse du modèle. Qui plus est, comme nous le verrons au prochain chapitre, les splines bénéficiant de propriétés intéressantes pour l'insertion/suppression de points, ce modèle est tout à fait adapté à la proposition d'un modèle mécanique adaptatif.

1.6 Conclusion sur les modèles

Comme ce chapitre a proposé une vue rapide de ce que la simulation physique implique, et des références pour les techniques principales. Le nombre de techniques impliquées dans la simulation physique est clairement très large. Maîtriser chacun des domaines est une chose difficile, et requiert un investissement bibliographique important. Il est clair que chaque modèle comporte ses avantages pour un contexte d'utilisation donné, mais aussi certains inconvénients pour d'autre. La question de savoir précisément quantifier ces avantages et limites, est en soi une question difficile. La question du coût de calcul de chaque modèle est souvent plus aisée à évaluer, et constitue souvent un frein à leur utilisation dans un contexte temps réel. Pour pallier à ce problème, certaines classes de modèles parmi ceux mentionnés dans ce chapitre sont associés dans la littérature à des techniques permettant d'adapter leur compléxité.

La problématique de ce chapitre, à savoir les modèles que la simulation physique temps réel utilise, a été assez artificiellement séparée de celle du prochain chapitre (ex: cas des objets à subdivision en modélisation géometrique, qui sont intrinsèquement multirésolution). Les deux chapitres, en effet, traitent des modèles que la simulation physique manipule, et de leur algorithmique. La raison de cette séparation est simple: l'adaptativité, comme on le verra dans le prochain chapitre, n'est pas présente dans la même proportion d'une classe de modèles à l'autre (ex: trés présente en modélisation, relativement peu en mécanique), et ils nous semble important, dans la problématique dont ce document traite, d'examiner cette question de manière separée. Le second chapitre de ce document a été préférentiellement, et de manière synthétique, orienté vers les modèles (dans les groupes que manipule la simulation physique, parmi l'affichage, la collision, la mécanique) dont on peut "adapter" la complexité, afin de coller de plus prés, au sein d'un même modèle à la perception hiérarchique que nous avons du monde, comme illustré dans l'introduction. Nous avons séparé cette argumentation sur les modèles adaptatifs, afin de les mettre en exergue, parmi les modèles que l'on manipule, de manière générale, en simulation.

Chapitre 2

Modèles adaptatifs pour la simulation physique temps réel

Le chapitre précédent a donné une vue d'ensemble des classes de modèles et d'algorithmique qui interviennent en simulation physique temps réel. Parmi ces modèles, certains bénéficient de techniques et propriétés leur permettant d'adapter leur complexité, que ce soit au niveau de la finesse de représentation, ou au niveau algorithmique. Ceci, dans le cadre général qui guide ce document, procure une optique tout-à-fait séduisante, car de telles propriétés permettent potentiellement d'adapter le temps de calcul ou le coût mémoire d'un objet.

2.1 Un mot sur la terminologie

Le titre de ce chapitre, ainsi que l'utilisation quie sera parfois faite dans le reste de ce document du terme "adaptatif", pourrait paraître impropre à certains, par rapport à quelques travaux mentionnés: les travaux sur les surfaces à subdivision, et leur gros bagage théorique, tous les outils de multirésolution par ondelettes sur les courbes et surfaces splines ne relèvent pas de la même sophistication, ni exactement des mêmes propriétés, que des techniques simples d'adaptation dynamiques de la résolution d'un modèle. cet usage du terme "adaptatif" a été fait par souci d'uniformité, et en l'absence de terme réellement unificateur: il existe clairement un flou autour de la définition de ce terme. Il n'existe en effet pas, à notre connaissance, de consensus sur la terminologie concernant les modèles à complexité variable: on parle souvent assez indifféremment d'objets adaptatifs, multirésolution, hiérarchiques. Les objets à "niveau de détail", sont parfois même inclus dans la même ambiguïté. Nous donnons ici quelques réflexions sur la question, et proposons de fait une classification (clairement subjective, se basant sur notre vision de la bibliographie du domaine) de ces termes, pour tenter d'éviter les confusions.

- modèles adaptatifs: il s'agit ici d'un terme regroupant des algorithmiques capable de réduire localement la résolution de la géométrie support, et à d'autre endroits, de l'augmenter, quand nécessaire. Il s'agit d'un terme très général. Les critères de simplification/raffinement peut être quelconques, et sans rapport direct avec une quelconque structure multirésolution de la géométrie support. L'algorithmique (quelle qu'elle soit) ne tire pas partie de la hiérarchie crée pour optimiser ses calculs, et considère toujours l'objet dans sa résolution

(non-uniforme) effective.

- modèles multirésolution: de notre point de vue, ce terme impose deux types de conditions: d'une part, une structure hiérarchique sur le modèle support, et d'autre part une algorithmique qui tire effectivement parti de la hiérarchie de la géométrie support pour optimiser un processus de calcul. Ce terme devrait rester associé à des algorithmiques tirant parti de structures hiérarchiques, semblables à ce que fourni la décomposition en ondelettes, ou les surfaces à subdivision (voir ci-dessous), i.e. des algorithmes se basant sur des géométries définies soit directement de manière hiérarchique, ou réorganisées par un mode de calcul plus ou moins complexe, *sans augmentation de la quantité globale de donnée*, de manière hiérarchique. Ce terme a été historiquement associé à des techniques utilisant une *représentation hiérarchique de la géométrie* pour optimiser un calcul donné: les courbes à multirésolution [FS94] proposent des solution pour éditer une courbe à divers niveau d'influence, en définissant des outils pour re-projeter sur une courbe modifiée à un certain niveau de résolution les "détails" (éléments des niveaux plus fins). De la même manière, les techniques dites de *multiresolution image querying* [JFS95] utilisent une décomposition par ondelettes des images pour accélérer la mise en correspondance.

- modèles hiérarchiques: ce terme semble associé aux géométries construites itérativement, telles les B-splines hiérarchiques [FB88], ou les surfaces à subdivision, ou bien encore aux structures crées, par utilisation de décomposition ondelette [Dau92], ou simplification hiérarchique, tel les normal meshes [GVSS00]. qui définissent de manière naturelle, pour un objet donné, un certain nombre de niveaux de représentations, ordonnés du plus grossier au plus fin. Cette terminologie n'est pas, à notre sens, associée comme les deux précédente à une algorithmique de traitement particulière; elle se distingue de plus de la notion de "niveau de détail" par le fait qu'on dispose d'une relation numérique entre les niveaux de représentations, et que l'obtention d'un niveau n de la hiérarchie se fait par la combinaison de la représentation à un niveau $n-1$ avec un certain nombre d'informations additionnelles.

- modèles à niveau de détail: les LOD sont aujourd'hui relativement courants dans les applications temps réel. Ils sont souvent une collection d'approximation plus ou moins fines pour une géométrie donnée. Il n'y a aucun prérequis de relation numérique quelconque entre les différentes approximations, ni, le plus souvent, d'algorithmique complexe (en dehors d'un simple affichage, où le niveau de détail choisi est sélectionné le plus souvent en fonction d'un critère de distance de visualisation).

Dans la mesure où les techniques dites à "niveau de détail" n'imposent pas d'homogénéité de représentation, nous les classons dans les techniques à multi-modèles (voir prochain chapitre). Nous traiterons ici des modèles "adaptatifs" au sens large, à défaut de terme unificateur.

2.2 Modèles géométriques adaptatifs

Nous classifions les quelques classes de résultats mentionnés ici de la même manière que nous l'avions fait dans le chapitre précédent, i.e. suivant la dimension des objets représentés.

2.2.1 objets linéiques

- splines: les B-splines possèdent une propriété les rendant particulièrement attractives pour l'adaptativité. L'insertion de noeud est le processus qui, partant d'un vecteur nodal donné, permet de calculer la manière dont on peut "raffiner" une courbe existante, c'est-à-dire la subdiviser de manière exacte, pour ainsi bénéficier de plus de points de contrôle que sur la courbe originale. Ce processus repose sur le résultat théorique concernant le fait que deux vecteurs nodaux, dont l'un est créé à partir de l'autre par insertion de noeuds, génèrent, pour un degré polynomial donné, deux espaces vectoriels B-splines imbriqués l'un dans l'autre [Pra85, Sch81] : une courbe générée à partir du vecteur nodal le plus "grossier" pourra être exactement exprimée dans l'espace B-spline correspondant au vecteur nodal le plus "fin".

  Formellement, considérons donc un premier espace de représentation B-spline, construit à partir d'un vecteur nodal $(u_0, \cdots, u_n)$, dont les fonctions d'influences, de degré polynomial k, sont notées ici F_j^0. Considérons ensuite un deuxième espace vectoriel B-spline construit à partir d'un vecteur nodal $(w_0, \cdots, w_N)$ englobant le vecteur précédent :

$$N > n \quad \text{et} \quad \forall i \in \{0, \cdots, n\},\ \exists j \in \{0, \cdots, N\} \,/\, u_i = w_j$$

  Les fonctions d'influences de degré k générées sont notées F_i^1. Le résultat théorique sus-cité correspond à l'existence de coefficients $\alpha_{i,j}^k$ tels que :

$$\forall i \in \{0, \cdots, n\},\ \forall t,\ F_i^0(t) = \sum_{i=0}^{N} \alpha_{i,j}^k F_j^1(t) \tag{2.1}$$

  Un algorithme célèbre, l'*algorithme d'Oslo*, permet d'évaluer ces coefficients $\alpha_{i,j}^k$: ceux-ci peuvent facilement s'évaluer via une formule de récurrence assez semblable à l'équation (1.2) définissant les fonctions d'influence B-splines :

$$\alpha_{i,j}^0 = \begin{cases} 1 \,\text{si}\, u_i \leq w_j < u_{i+1} \\ 0 \,\text{sinon} \end{cases}$$

$$\text{et} \tag{2.2}$$

$$\forall r = 0, \cdots, k-1,\ \alpha_{i,j}^{r+1} = \frac{w_{j+r} - u_i}{w_{i+r} - u_i} \alpha_{i,j}^r + \frac{u_{i+r+1} - w_{j+r}}{u_{i+r+1} - u_{i+1}} \alpha_{i+1,j}^r$$

  De la même manière que pour la définition des B-splines, les termes impliqués sont considérés comme nuls si leurs dénominateurs le sont. Certains résultats existent [BBB87] sur ces coefficients $\alpha_{i,j}^k$:

$$\forall j,\ \sum_i \alpha_{i,j}^k = 1$$

$$\forall (i,j),\ \alpha_{i,j}^k \geq 0$$

$$\forall h \notin \{i-k, \cdots, i\},\ \alpha_{h,j}^k = 0 \qquad \text{avec} \qquad i \,/\, u_i \leq w_j < u_{i+1}$$

  La dernière de ces trois propriétés est probablement la plus intéressante car elle nous garantie que le processus d'insertion de noeud ne modifiera que localement les points de contrôle de la courbe considérée. L'expression donnant les points de contrôle raffinés

en fonctions des plus grossiers découle naturellement de (2.1) et de la définition (1.1), en égalisant les deux expressions analytiques d'une même courbe, dans chacun des deux espaces vectoriels.

- subdivision: les courbes à subdivision donnent, de manière immédiate, une vision hiérarchique du modèle. L'utilisation hiérarchique est en fait l'utilisation la plus fréquente de ces modèles: trés rares sont les applications qui, comme décrit au premier chapitre, considèrent uniquement l'objet continu sous-jacent: l'intéret pratique de ces modèles et de créer une hiérarchie d'objet simples (linéaires par morceaux), et donc facilement manipulables. La subdivision garanti que le raffinement de la courbe produit une version lissée de celle-ci.

- courbes multirésolution à base d'ondelettes: ces outils, servant à construire, à partir d'une courbe, une représentation hiérarchique, ont fait l'objet d'une étude approfondie en géométrie, en compression et en analyse d'image depuis une quinzaine d'année [SDS96]. Par le biais de projection de l'objet initial sur un espace fonctionnel, on crée la hiérarchie de représentation par le biais de projection successives, sur une suite d'espaces fonctionnels imbriqués[Mal88]. Ces outils ont l'avantage de procurer de manière relativement simple une structure hiérarchique à partir d'un objet donné. Elles sont, à notre connaissance, le meilleur point d'entrée pour les algorithmes multirésolution, tirant parti de la hiérarchie d'un modèle pour optimiser leur calcul. Toutefois, dans le cadre du temps réel, leur utilisation est pour le moment assez délicate, dans la mesure où l'on doit en pratique choisir entre des fonctions de représentation pour lesquelles on dispose d'expression paramétrique, et de filtre à réponse impulsionelle finie (notamment pour la phase d'analyse [Gri99]): l'un comme l'autre sont utiles pour manipuler simplement la représentation continue sous-jacente, et, en cas d'absence, limitent fortement l'efficacité d'un algorithme potentiellement applicable sur les courbes.

2.2.2 modèles surfaciques

- splines: ces outils, de par la propriété des splines 1D, et le fait que la plupart des surfaces splines sont construites par produit tensoriel, bénéficient de manière naturelle de construction hiérarchique: toutefois, l'approche simple, à cause de la construction tensorielle, augmente de manière non-optimale la quantité de donnée: les raffinements, en modélisation géometrique sont souvent des éléments de construction locale, non-tensoriels. Pour pallier ce problème, les splines hiérarchiques [FB88] fournissent un moyen accessible pour le temps réel de manipuler un modèle spline hiérarchique. Les T-splines disposent également de technique adaptatives [SCF$^+$04].

- subdivision: les surfaces à subdivision constituent, à l'heure actuelle, le domaine de prédilection pour la manipulation de modèles surfaciques hiérarchiques. Leur manipulation pour l'affichage est ces dernières années en pleine expansion [SJP05].

- ondelettes construites sur des topologies arbitraires: ces techniques permettent, à partir de maillages quelconques, de définir une hiérarchie de représentation. Ces techniques [SS95] se basent sur un couplage entre la subdivision (ce qui implique au passage, pour un maillage donné, de passer par une étape de remaillage, pour mettre la topologie du maillage en conformité avec ce que donne la subdivision) et une famille d'ondelettes biorthogonales construites par factorisation (théorème du Lifting Scheme[Swe95])

- surfaces reconstruites par nuage de points: les techniques de reconstruction de surfaces tirent aussi, depuis peu de temps, parti des structures hiérarchiques. l'utilisation de partitionnement spatial adaptatif (typiquement octree) permet de reconstruire localement les aretes vives [OBA+03], en utilisant un critère local pour détecter ses structures: une combinaison de type "partition of unity" est ensuite utilisée pour assembler les diverses reconstructions locales avec un niveau de continuité satisfaisant. Il est à noter que certaines techniques utilisent la structure hiérarchique pour reconstruire la surface de manière, elle aussi, hiérarchique, se rapprochant alors, dans la terminologie proposée, d'une réelle reconstruction multirésolution [AA03].

2.2.3 modèles volumiques

Il y a relativement peu de travaux portant sur les représentations volumiques hiérarchiques, pour la classe utile dans le cadre de la simulation physique. On peut toutefois noter [LH91], qui propose une estimation adaptative de la densité locale, pour en déduire une procédure d'affichage, par une technique de combinaison de gaussiennes 2D. La structure d'octree fournit également un moyen simple, couramment utilisée, pour définir une hiérarchie volumique.

2.3 Modèles de collision adaptatifs

Comme décrit dans le premier chapitre, la détection de collision est encore, dans le cas général et au meilleur des techniques actuelles, un calcul très coûteux. On trouve, dans la littérature sur la question, quantité de travaux pouvant se ramener à une vision adaptative de la détection de collision:

- utilisation de volumes englobants: il s'agit d'une méthode trés classique en détection de collision. Ces volumes englobants (k-DOP ou autres) peuvent être facilement organisés en hiérarchie [KHM+98], permettant ainsi d'obtenir un modèle de collision pouvant être plus ou moins précis. La difficulté principale est que, dans le cas déformable, la hiérarchie doit être mise à jour. Les objets peu déformables bénéficient toutefois de techniques de mise à jour efficaces [JP04].

- collision interruptible: [OD01a] propose une méthode de détection de collision, basée sur une hiérarchie de sphères [Hub95], qui permet une détection de collision pouvant être interrompue. Une étude a, dans le cadre de son travail, été menée sur l'impact perceptuel d'une dégradation volontaire du niveau de réponse à la collision.

2.4 Modèles mécaniques adaptatifs

L'adaptivité intervient de manière inégale sur la mécanique: de manière forte sur les lois mécaniques proprement dites, de manière assez faible sur l'intégration numérique, et, à notre connaissance, de manière inexistante sur les gestions de contraintes (nous n'abordons d'ailleurs pas la question des contraintes adaptatives).

2.4.1 modèle mécanique

On peut dégager quelques classes de méthodes adaptatives, même si la bibliographie, dans ce domaine, est à notre perception encore relativement réduite

- modèle masse-ressort adaptatifs: [HPH96] propose une version primitive de mécanique adaptative, par le biais du raffinement local d'un système masse-ressort. Le critère de division est basé sur une estimation discrète de la courbure locale de la surface, et, dans le cas de raffinement, une nouvelle masse est insérée, en remplacant les ressorts concernés par deux ressorts, de raideur deux fois plus grande: la masse globale du système n'est pas conservée, et le système, si le raffinement est reproduit récursivement, tends rapidement vers un système raide, causant des difficultés d'intégration.
- Gilles Debunnes, dans sa thèse [Deb00] propose deux méthodes de méchaniques adaptatives, une première basée sur un octree (combiné à un système de particule, permettant de s'affranchir du problème de T-jonction, voir [DMG05]), la deuxième basée sur un système tétrahédrique, ou plusieurs niveau de résolution sont dynamiquement combinés, pour proposer un modèle mécanique adaptatif.
- certains travaux utilisent le cadre théorique de la subdivision pour proposer des résultats de mécanique adaptative: [GKS02] défini une approche élément fini adaptative d'ordre supérieur; [JP03] défini un cadre rigoureux pour la simulation d'objet déformable, se basant sur une version multirésolution du processur de reconstruction de surface à base de fonction de Green. Ces techniques sont potentiellement celles qui permettront la mécanique la plus rigoureuse, car se basant sur la structure théorique la plus fiable. Elles sont toutefois assez délicate à mettre en oeuvre, et nécessite un investissement théorique important.
- [RGL05] propose une technique de simulation adaptative de corps rigide articulé: la technique décrite, en fonction des effort exercés sur le système, sélectionne les degrés de libertés significatifs, en interpolant les autres.

2.4.2 méthodes d'intégration

Il existe, à notre connaissance, relativement peu de méthodes pouvant être qualifiées d'adaptatives pour l'intégration numérique. Toutefois, les méthodes à pas de temps variables proposent des méthodes proches de cette philosophie: comme mentionné dans le premier chapitre, le problème essentiel que l'intégration numérique doit gérer est l'instabilité, qui implique généralement de réduire le pas de temps d'intégration. Certaines méthodes [DDCB01] proposent des outils pour subdiviser un pas de temps en sous-pas, et effectuer, au sein d'un pas de temps donné, une intégration en plusieurs étapes.

2.5 Contributions

2.5.1 rendu haute performance de cylindre généralisé [GM03]

Ce travail s'est inscrit dans le cadre de l'ARC SCI, en collaboration avec le projet EVASION (responsable Marie-Paule Cani) et l'IRCAD (Institut de recherche sur le Cancer de l'Appareil

Digestif, Strasbourg). Le problème initial était de savoir comment afficher efficacement l'appareil intestinal, sachant que géometriquement parlant, il s'agit d'un modèle 1D hautement déformable. A partir de cette réflexion s'est dégagé la technique présentée dans cet article, géneralisée à un sous-ensemble de la classe des cylindres généralisés [AB76]. Cette technique utilise une extension récente des cartes 3D du marché, en réduisant significativement la quantité d'information transitant par le bus vidéo: Les informations nécessaire à la carte pour mener l'affichage à bien est uniquement composées d'une suite de matrices de transformations. Cette technique est combinée à un critère d'évaluation de la courbure splines, utilisant la propriété de régularité de ces dernières, et ramenant l'estimation de la courbure sur un segment à une évaluation angulaire sur la configuration de point de contrôle locale. La carte vidéo interpole ensuite les matrices de transformations pour effectuer une déformation de la primitive d'affichage, créant ainsi un résultat visuel d'une bien meilleure continuité, en améliorant de manière très importante les performances. Les performances graphiques étant largement au dessus de ce que dont nous avions besoin dans le simulateur, s'est posé de manière très naturelle la question de savoir comment régler le modèle, pour obtenir un affichage de la meilleure qualité possible, en gardant un nombre d'image par seconde raisonnable. Cette question est assez proche de celle à laquelle répond la notion de *fonction de coût* [FS93], utilisée depuis quelques années déjà pour les techniques d'affichage à base de niveau de détail. Nous avons proposé, associé à ce modèle, un module de contrôle externe qui, dynamiquement, est capable d'adapter la résolution du modèle d'affichage à une contrainte de performance d'affichage pre-imposée. Ce module de contrôle présente l'avantage important de pouvoir s'adapter à des contraintes d'affichages variables, notamment de pouvoir réduire la complexité du modèle, si d'aventure, d'autre routines d'affichage devaient être combinées.

2.5.2 Modèle déformable spline 1D adaptatif[LGMC05]

Ce travail étend les résultats de [7, 8, 11] et propose une dérivation du modèle capable d'adapter dynamiquement sa configuration géométrique à l'utilisation qui en est faite. Ce résultat a été motivé par le fait que, pour mener à bien une opération de serrage de noeud, on doit bénéficier localement d'une bonne résolution, sinon le modèle mécanique est assez vite limité dans la phase de serrage. Un critère simple d'évaluation de courbure locale (le même que celui utilisé dans [GM03]) a été utilisé pour, combiné à l'algorithme d'insertion de points d'Oslo (voir section 2.2.1), pouvoir raffiner localement, là où nécessaire, la configuration géométrique de la courbe B-spline. Les termes énergétiques, exprimés dans le cadre B-spline non-uniforme, sont recalculés dynamiquement pour assurer la continuité de la simulation mécanique. Des techniques sophistiquées de simplifications de courbes B-splines existent [LM87], mais, dans le cadre du temps réel, nous nous sommes dans ce travail limité au même évaluateur de courbure, cette fois utilisé pour évaluer les zones de courbure faible. Ce critère est très rapide à évaluer et permet, comme illustré dans la vidéo associée à l'article, de serrer correctement un noeud, d'obtenir à la fin une configuration géométrique de noeud serré correcte, en gardant une complexité géométrique quasi-constante. Il est à noter un effet de bord extrêmement intéressant de la technique nécessaire à l'insertion de noeud dans un fil spline dynamique: un résultat B-spline classique est que, sous couvert d'augmenter suffisament la multiplicité d'un noeud, on peut descendre à une continuité géométrique C^{-1}. Ceci, transposé au monde de la simulation mécanique, et sous couvert (ce qui est le cas avec les travaux décrit dans cet article) de disposer des règles d'adaptation de la mécanique à une insertion de noeud, signifie que l'on peut, en ayant une procédure d'affichage adaptée, couper le fil spline à une valeur de paramètre totalement arbitraire, et créer en fait autant de morceaux que l'on souhaite.

2.5.3 Un modèle volumique déformable à temps constant [DMG05]

Ce travail fourni un modèle volumique déformable générique, pouvant adapter totalement son temps de calcul de simulation (collision + mécanique) à une contrainte de temps de calcul pré-imposée. La technique utilisée pour la simulation est une simulation type éléments fini hexaédrique (utilisés ensuite comme grille FFD pour la déformation du modèle proprement dit), adaptée sur une structure de type octree. L'octree est rendu linéaire par une numérotation particulière, le codage de Morton [LS00]. Tant pour la collision [OD01b] que pour la mécanique [DDCB01], des résultats existent qui proposent déjà de l'adaptatif. La contribution essentielle de cet article est d'avoir intégré de l'adaptivité dans toutes les étapes ou cela est possible, et de proposer une boucle de contrôle globale qui permette d'optimiser la résolution (tant en collision qu'en mécanique, de manière indépendante) pour essayer de respecter, moyennant une certaine tolérance, un temps de calcul. L'approche pour la définition de la boucle de contrôle est différente de celle utilisée pour [GM03]: ici, des tests de simplification/raffinement sont effectués à la volée pour estimer le coût de ces opérations, le nombre et la nature de ces opérations étant ensuite déterminé à parti de ces mesures. Ce type de boucle de contrôle est à priori plus proche d'une (pour le moment) hypothétique boucle de contrôle générique, dans la mesure où elle tire bien moins parti d'une connaissance fine du modèle qu'elle contrôle: par comparaison, le système de contrôle proposé dans [GM03] est, lui, trés intimement basé sur une bonne connaissance des paramètres du modèle controlé, et de leur influence sur la rapidité d'affichage.

2.6 Conclusion

Ce chapitre a proposé une vue des différents modèles et techniques adaptatives (ou multirésolution) qui emergent en simulation physique temps réel. Ces techniques permettent, dans une certaine mesure, d'adapter le cout d'un modèle en changeant sa complexite, dynamiquement. Dans certains cas (comme le petit exemple d'une balle animée, cité dans l'introduction de ce document) les limites du modèle lui-meme devront être repousée: on pourra alors chercher a changer dynamiquement de modèle et de domaine de validité, pour avoir un comportement dde l'objet simulé plus souple, et une réelle adaptation dynamique au contexte d'utilisation. Le prochain chapitre traite de cette problématique, la plus générale de ce document, d'objets multi-modèle.

Chapitre 3

Multi-modèle pour l'animation physique temps réel

Il est à noter, dans le chapitre précédent, que deux des trois contributions présentées proposent une fonctionnalité externe au modèle, qui permet d'enrichir ce dernier en lui permettant de garantir, dans une certaine mesure, le respect de contraintes de performances imposées de manière externe au modèle. Ceci peut être vu comme un exemple, certes encore primitif, d'objet physique "intelligent", capable de proposer à la simulation globale des garanties de performances. Dans cette optique, le centre d'intérêt principal ne devient plus le modèle lui-même, mais les fonctionnalités que l'objet va proposer: on dépasse donc le niveau du modèle, pour considérer un objet simulé comme une entité complexe, pouvant de manière ultime gérer sa représentation, comme c'était le cas dans le chapitre précédent en changeant la complexité des modèles impliqués, ou bien encore, en adaptant dynamiquement les modèles eux-meme au contexte d'utilisation. Ce chapitre traite des techniques pouvant être associée à cette dernière idée.

3.1 Etat de l'art sur le multi-modèle

La bibliographie pouvant être reliée à la thématique de ce chapitre est clairement plus réduite que pour les chapitres précédents: nous citons ici les quelques travaux portés à notre connaissance, qui peuvent être reliés à la thématique décrite dans ce chapitre.

3.1.1 LOD pour l'animation

Jessica Hodgins a étendu la notion de niveau de détail à l'animation (non-nécessairement physique) [CH97]: Dans cet article, est proposé un modèle d'animation simple proposant trois représentations, à partir duquel une évaluation du gain potentiel en temps de calcul est présenté. Les critères de choix de représentation se basent sur une évaluation du type de représentation à utiliser (certaines peuvent gérer une collision, d'autre non) combiné à des critères de visibilité/distance, classiques depuis les travaux de Funkhauser et Sequin [FS93] sur la notion de fonction de coût pour les niveaux de détails géométriques. Il est à noter que ce travail n'a pas été, à notre connaissance étendu à des simulations plus sophistiquées, et n'apporte pas d'éléments de réponses sur la conservation des critèrés dynamique lors du passage d'un niveau à l'autre.

3.1.2 multi-modèle mécanique

Quelques travaux, s'il sont moins généraux que le travail d'Hodgins, mixent déjà plusieurs types de représentations mécaniques:

- utilisation du corotationnel pour les éléments finis: comme mentionné au premier chapitre, on peut associer, pour ne pas avoir à fixer un point de l'objet, à la simulation déformable une composante corotationnelle [THMG04], qui est une composante intrinsèquement rigide du mouvement. Cette technique fait, de manière implicite, cohabiter deux représentations: une représentation déformable pour les déformations internes, et une représentation rigide, pour le mouvement global de l'objet.
- une approche mixte entre déformable et rigide: [MHTG05] présente une méthode originale de simulation d'objet déformable, tirant encore plus partie de la dualité rigide/déformable: un objet est défini à partir d'un ensemble de particules. A partir d'une intégration simple du mouvement de ces particules, une position du cadre rigide de référence est évaluée. A partir de cette nouvelle position du cadre de référence rigide, les positions des particules sont corrigées pour tenir compte à la fois de la position donnée par la simulation mécanique, et de la position si l'objet était totalement rigide. Cette technique évite ainsi de gérer une vraie mécanique déformable, gagnant ainsi en rapidité. La rigueur mécanique est difficile à atteindre, mais la plausibilité est là.
- Dans le cas de la simulation de l'explosion d'objets [MTG04] utilise une simulation rigide, et, au moment de l'impact, pour un pas de temps, utilise une mécanique déformable pour déterminer les points de fracture.
- [JJ03] présente une technique, permettant de faire de l'adaptatif pour les modèles peu déformables: une structure mixte couplant déformable et rigide, liés entre eux (ex: en une chaîne pour l'exemple de la poutre) est utilisée pour approximer un objet. Cette technique est un cas intéressant d'objet multi-représentation de manière *interne* au modèle, couplant, à une date donnée, plusieurs modèles mécaniques.

3.1.3 multi-modèle pour la collision

La notion de *pipeline* de collision [Zac01] est une notion trés proche de la philosophie du multi-modèle: une suite d'algorithmique, allant des plus rapides (et moins précises), aux plus complexes (et précises) est construite. Pour gérer les auto-collisions d'un objet déformable, le pipeline continue la hiérarchie au sein de l'objet, pour construire des groupes de primitives [GKJ+05], ce qui entre tout à fait dans l'optique multi-modèle.

3.2 contributions: une proposition d'architecture adaptée au multi-modèle [DGC04]

Dans la mesure où l'on souhaite permettre que les systèmes mécaniques que l'on considère pour une simulation donnée puisse bénéficier de règles de décision leur permettant de changer dynamiquement de résolution, ou d'une manière plus générale changer de modèle de représentation, la question de savoir comment combiner entre eux des systèmes mécaniques simulés, en tentant

de minimiser les hypothèses concernant l'algorithmique de chaque système, se pose de manière naturelle. Dans sa thèse, J. Dequidt a proposé une architecture allant, autant que possible, vers une indépendance des systèmes mécaniques qui constituent une simulation: [DGC04] propose une architecture où chaque système mécanique est considéré comme un agent indépendant. Afin de permettre a chaque objet de potentiellement adapter librement son pas de temps de calcul, l'architecture a été testée sans imposer de contraintes de synchronisation forte entre les objets: les tests effectués montrent qu'une telle relaxation est possible, sans introduire d'erreur significative, dans un certains nombres de cas. Il est à noter que les quelques exemples d'objets "intelligents" illustré dans l'article (objet adaptant dynamiquement leur pas de temps de calcul au contexte d'interaction, et objets interagissant, disposant chacun de leur propre méthode d'intégration numérique) ne sont pas, en soi, des méthodes sophistiquées, ni délicates à mettre en place: toutefois, le fait d'envisager les systèmes mécaniques comme étant des systèmes "clos", autonomes, permet de les associer facilement (tant intellectuellement que logiciellement) à des fonctionnalités qui ne sont pas au coeur de la recherche en simulation classique, mais qui ajoutent de la sophistication aux objets simulés. C'est cet aspect que nous considérons comme avant tout intéressant dans cette approche. Il est à noter que ce travail a eu une dérivation intéressante: dans la mesure où cette architecture nous a permis de valider le fait que dans certains cas, l'interaction entre systèmes mécanique tolère un relatif décalage dans les dates de simulation, nous avons pu proposer un système qui permet de réaliser des manipulations de simulation physique de manière collaborative [15], via le réseau: cette architecture ne repose pas sur un classique système client/serveur, mais sur un système de synchronisation lâche qui, de manière asynchrone et suivant les performances du réseau, tente de corriger les simulations des divers postes pour réduire le décalage constaté entre les simulations. Ainsi, chaque utilisateur conserve une interactivité forte, indépendant des performances réseau, avec la scène. Dans le cas de bonnes performances réseau, on arrive avec cette architecture à effectuer des manipulation réellement collaborative (ex: relever à deux utilisateurs un tissu).

3.3 Conclusion

La notion de multi-modèle est encore une thématique très jeune en simulation physique, et n'apparaît encore que de manière embryonnaire dans la bibliographie. La conclusion globale de ce document présente quelques pistes, à moyen et long terme.

Chapitre 4

Conclusion et axes de recherches

Ce document a tenté de donner un aperçu de mon activité de recherche, en mettant en exergue le fil conducteur de celle-ci: aller vers une simulation physique temps réel multi-modèle, ou chaque système mécanique impliqué pourrait, suivant son contexte d'utilisation, adapter ses représentations géométrique et mécanique. Il a également tenté de mettre en lumière les points où l'activité de recherche est la plus active, parmi les problèmes que la simulation physique temps réel adresse; la même description tente de mettre l'accent également, de manière duale, sur certains aspects pour lesquels il existe encore peu de résultats, et les problèmes encore délicats à gérer en temps réel. Pour évaluer les axes de recherches que la problématique présentée peut générer, on peut, de manière immédiate, livrer les éléments de réflexion suivants:

- à propos de la recherche sur les modèles en général:
  - Un effort important à porter est, selon nous, sur la gestion en temps réel de modèles mécaniquement rigoureux, pour coller de prés, dans la gamme d'objets simulés pour le moment accessibles en temps réel, pouvoir proposer des simulations réellement fidèles à la réalité, ou du moins aussi fidèles que possible à celle-ci.
  - certaines gammes de simulation (ex fluides, interaction fluide-solide) sont, à notre connaissance, non-gérables en temps réel. Leur gestion en temps réel constitue de manière évidente un challenge considérable.
  - Il est à noter que l'on ne tire probablement pas suffisamment parti d'artifices d'affichage 3D, pour pallier aux limites de calcul actuelles que la mécanique rencontre (les résultats de mécanique meshless temps réel, ou de déformation volumique générique par FFD en sont toutefois de premiers exemples): ceci constitue également une base de réflexion.
  - Les cartes 3D sont de plus en plus complexes, et offrent une unité de calcul de plus en plus attrayante (le moteur de simulation physique Havoc a d'ailleurs très récemment annoncer supporter, dans sa prochaine version, de la simulation physique sur GPU). La question de savoir si l'utilisation du GPU pour autre chose que de l'affichage est une activité de recherche proprement dite est une question sensible dans la communauté scientifique, mais il reste néanmoins que cette acquisition de compétence technique, si elle ne relève pas d'un apport théorique pérènne, permet semble-t-il d'alimenter

une activité de publication de bon niveau [BFGS03, SJP05, BS05], et permettra de manière incontestable de tirer au mieux parti des architectures modernes, pour améliorer les simulations physiques temps réel accessibles sur les machines actuelles.

- à propos de la recherche sur les modèles adaptatifs:
  - Les travaux sur le fil, peuvent à eux seuls, amener une réflexion de fond sur cet aspect: il reste encore beaucoup de travail, tant théorique que sur la structure logicielle, pour disposer d'un module de fil chirurgical qui soit réellement intégrable aisément dans un simulateur, et compatible, par le biais de gestion dynamique de contrainte, avec des opérations de sutures, de découpe de fil, etc...
  - les opérations de découpes peuvent, elles aussi, alimenter la recherche sur ce domaine, dans la mesure où le contrôle précis de la trace de découpe est une question difficile à résoudre, dans un contexte temps réel.
  - Il serait intéressant de proposer une généralisation du processus de contrôle présent dans [GM03], ainsi que la version, un peu plus générique, de [DMG05]: un tel outil permettrait alors, à partir du moment où un modèle est "réglable", de disposer, sans aucune connaissance spécifique du modèle en question, d'un module lui permettant d'intégrer des applications à performances garanties.

- à propos de la recherche sur le multi-modèle proprement dit:
  - le cadre logiciel: le projet SOFA est un projet assez ambitieux d'architecture dédiée à la simulation physique temps réel, pour le moment collaboration entre l'INRIA (projets EPIDAURE, EVASION et ALCOVE), et le Simulation Group du CIMIT (MIT/Harvard Hospital, Boston, USA). Cette plateforme est destinée aux laboratoires de recherche sur ce thème, pour tenter de fournir une plateforme d'accueil aux algorithmes les plus sophistiqués, et fournir ausi quand nécessaire une base de développement. La première version de cette plateforme devrait être rendue publique courant Janvier 2006, lors de la conférence MMVR. La question de réussir à proposer une plateforme suffisamment fonctionnelle pour être utilisable, et suffisamment fédératrice pour ne pas rester stérile, est une question extrêmement difficile. Toutefois, SOFA a, dans sa version actuelle, su trouver l'intersection d'intérêt de 4 équipes de recherches, chacune venant avec sa sensibilité et son expérience propre. Le fait d'avoir la chance d'être au coeur de la naissance d'un projet d'une telle ampleur est une aventure humaine extrêmement enrichissante, et nous sommes pour le moment optimiste dans le devenir de ce projet: à titre personnel, je suis intimement convaincu que cette plateforme sera une plateforme d'accueil idéale pour les objets multi-modèles à venir, dans la mesure où beaucoup des concepts d'encapsulation fonctionnels, issus de la thèse de J. Dequidt, ont été repris dans l'architecture de base de cette plateforme.
  - la question de la conversion entre modèles est une question cruciale, pour espérer disposer de tels objets. Tant dans le domaine géométrique, que celui de la collision, ou de la mécanique, la question de la conversion d'un modèle à l'autre alimente depuis de nombreuses années, et alimentera encore, l'activité de publication de recherche; certaines conversions, dans le domaine mécanique notamment, où peu de chose existent sur la question, semblent accessibles assez rapidement. D'autres se heurteront certainement au limites de chaque modèle, et donneront sans nul doute nombre de problèmes aussi difficiles qu'intéressants.

- Au delà de la question de savoir comment passer d'un modèle à l'autre, la question de savoir *quand* le faire est, elle aussi, une question importante, et mériterait une étude plus approfondie que ce à quoi elle a eu droit jusqu'à présent [OHM+04]. Il est toutefois nécessaire de bénéficier d'une petite base de travail stable pour étudier cette question (i.e. disposer déjà d'un ensemble de modèles facilement "interchangeables").
- Certains aspects, simplement descriptibles, alimenteront eux aussi la recherche sur le multi-modèle. Dans le cadre de la découpe, le changement de connexité d'un objet est encore trés mal géré par les simulateurs actuels: on ne sait pas detecter, en temps réel, cette séparation complète d'un objet en deux partie. Vue de l'optique multi-modèle, cette opération peut être vue comme un système mécanique qui, à la suite de l'opération de découpe, se scinde en deux système autonomes: leurs simulations (sous couvert qu'il n'y ait pas d'interaction entre les deux) n'ont plus alors de raison d'être dépendantes l'une de l'autre, sous couvert que l'on sache comment gérer cette question de manière souple logiciellement, et efficace algorithmiquement.
- Un autre type de manipulation inspire la réflexion inverse: un cylindre, emboîté dans un autre, peut certes être simulé par la physique, mais la contrainte est telle que les deux objets sont immobiles l'un pour l'autre, et ne constituent de fait plus, pour la simulation, qu'un seul système mécanique. La question de la co-existence de la simulation physique avec l'assemblage, et les optimisations qui pourraient en découler, est, elle aussi, une question intéressante.
- Dans la simulation existent un certain nombres de phénomènes que l'on sait simuler, mais que l'on pourrait tout à fait *émuler*, dans un contexte ou le besoin de précision, ou d'interaction, est moindre. La question de savoir comment passer de l'un à l'autre, et là aussi, dans quel type de contexte, est une question à notre connaissance totalement ouverte.

Bibliographie

[AA03] Anders Adamson and Marc Alexa. Ray tracing point set surfaces. Proceedings of Shape Modeling International 2003, pages 272-279, 2003.

[AB76] G. J. Agin and T. O. Binford. Computer description of curved objects. In *IEEE Trans. Computers 25(4): 439-449*, 1976.

[AEV03] Nicolas Aspert, Touradj Ebrahimi, and Pierre Vandergheynst. Non-linear subdivision using local spherical coordinates. *Comput. Aided Geom. Des.*, 20(3):165–187, 2003.

[AN] Alexis Angelidis and Fabrice Neyret. Simulation of smoke based on vortex filament primitives. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

[Bar91] David Baraff. Coping with friction for non-penetrating rigid body simulation. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 31–41, New York, NY, USA, 1991. ACM Press.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34, New York, NY, USA, 1994. ACM Press.

[Bau72] J. Baumgarte. Stabilization of constraints and integrals of motion. Computer Meth. Appl. Mech. Eng. 1, 116., 1972.

[BBB87] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, San Fransisco, CA, 1987.

[BBCK03] D. Blandford, G. Blelloch, D. Cardoze, and C. Kadow. Compact representations of simplicial meshes in two and three dimensions, 2003.

[BEH02] Marshall Wayne Bern, David Eppstein, and Brad Hutchings. Algorithms for coloring quadtrees. *Algorithmica*, 32(1):87–94, January 2002.

[BFGS03] Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröoder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22(3):917–924, 2003.

[BNC96] Morten Bro-Nielsen and Stephane Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, 1996.

[Boy79] John W. Boyse. Interference detection among solids and surfaces. *Commun. ACM*, 22(1):3–9, 1979.

[BS05] Tamy Boubekeur and Christophe Schlick. Generic mesh refinement on gpu. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 99–104, New York, NY, USA, 2005. ACM Press.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM Press.

[BWK03] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[BYM05] Nathan Bell, Yizhou Yu, and Peter J. Mucha. Particle-based simulation of granular materials. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 77–86, New York, NY, USA, 2005. ACM Press.

[Cam97] S. Cameron. Enhancing GJK: computing minimum and penetration distances between convex polyhedra. In *Int. Conf. Robotics & Automation*, April 1997.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM Press.

[CH97] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 1–8. Canadian Human-Computer Communications Society, 1997.

[Coh00] Albert Cohen. *Wavelet methods in numerical analysis, Handbook of Numerical Analysis, vol. VII, P.G.Ciarlet and J.L.Lions eds., Elsevier, Amsterdam.* 2000.

[Cot97] S. Cotin. *modèles anatomiques déformables en temps réel: Apllication à la simulation de chirurgie avec retour d'effort.* PhD thesis, Université de Nice Sophia-Antipolis, France, 1997.

[Cou80] J. Courbon. *Théorie des poutres.* Techniques de l'ingénieur, dossier C2010, August 1980.

[Cox72] M. G. Cox. The Numerical Evaluation pf B-Splines. In *J. Inst. Mathematics and Applications*, volume 10, pages 134–149, 1972.

[D'A01] D. D'Aulignac. *Modélisation de l'Interaction avec Objets Déformables en temps-réel pour des simulateurs chirurgicaux.* PhD thesis, INPG, Grenoble,, France, 2001.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets*. CBMSF-NSF Regional Conf. Series in Appl. Math. w, vol. 61, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[dB72] C. de Boor. On Calculating with B-Splines. *J. Approximation Theory*, 6(1):50–62, July 1972.

[DC94] Mathieu Desbrun and Marie-Paule Cani. Highly deformable material for animation and collision processing. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, septembre 1994. Published under the name Marie-Paule Gascuel.

[DC95] Mathieu Desbrun and Marie-Paule Cani. Animating soft substances with implicit surfaces. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, volume 29 of *Annual Conference Series*, pages 287–290. ACM SIGGRAPH, Addison Wesley, aug 1995. Los Angeles, California, published under the name Marie-Paule Gascuel.

[DCA99] Hervé Delingette, Stéphane Cotin, and Nicholas Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *CA '99: Proceedings of the Computer Animation*, page 70, Washington, DC, USA, 1999. IEEE Computer Society.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 31–36, New York, NY, USA, 2001. ACM Press.

[Deb00] Gilles Debunne. *Animation multirésolution d'objets déformables en temps-réel, Application à la simulation chirurgicale*. PhD thesis, INPG, Grenoble, France, 2000.

[DGC04] Jérémie Dequidt, Laurent Grisoni, and Christophe Chaillou. Asynchronous interactive physical simulation. INRIA Tech. report n. RR-5338. http://www.inria.fr/rrrt/rr-5338.html, 2004.

[DMG05] J. Dequidt, D. Marchal, and L. Grisoni. Time-critical animation of deformable solids: Collision detection and deformable objects. *Comput. Animat. Virtual Worlds*, 16(3-4):177–187, 2005.

[Dur04a] C. Duriez. *Contact frottant entre objets déformables dans des simulations temps-réel avec retour haptique*. PhD thesis, Université d'Evry, 2004.

[Dur04b] C. Duriez. *Contact frottant entre objets déformables dans des simulations temps-réel avec retour haptique*. PhD thesis, Université d'Evry, France, 2004.

[DZ00] P. Schröder D. Zorin. Subdivision for modeling and animation. Siggraph Course Notes, 2000.

[EC09] F. Cosserat E. Cosserat. *Théorie des objets déformables*. 1909.

[EGK88] D. W. Johnson E.G. Gilbert and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, RA-4:193–203, 1988.

[Far92] G. Farin. *Courbes et surfaces pour la CGAO*. Masson, 1992.

[Fau96] François Faure. An energy-based approach for contact force computation. In Jarek Rossignac and François Sillion, editors, *Computer Graphics Forum (Proc. of Eurographics)*, volume 16, pages 357–366, Poitiers, France, Sep 1996.

[Fau98] François Faure. Interactive solid animation using linearized displacement constraints. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, pages 61–72, 1998. ISBN 3-211-83257-2.

[FB88] D. R. Forsey and R. H. Bartels. Hierarchical b-spline refinement. In *Computer Graphics Proceedings SIGGRAPH '88*, Annual Conference Series, pages 205–212. ACM, 1988.

[Fed02] Ronald P. Fedkiw. Coupling an eulerian fluid calculation to a lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175(1):200–224, 2002.

[FLA+05] Laure France, Julien Lenoir, Alexis Angelidis, Philippe Meseure, Marie-Paule Cani, François Faure, and Christophe Chaillou. A layered model of a virtual human intestine for surgery simulation. *Medical Images Analysis, Elsevier Sciences*, 2005.

[For03] Clément Forest. *Simulation de chirurgie par coelioscopie : contributions à l'étude de la découpe volumique, au retour d'effort et à la modélisation des vaisseaux sanguins*. PhD thesis, École Polytechnique, March 2003.

[FS93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 247–254, New York, NY, USA, 1993. ACM Press.

[FS94] A. Finkelstein and D. H. Salesin. Multiresolution curves. In *Computer Graphics Proceedings SIGGRAPH '94*, Annual Conference Series, pages 261–268. ACM, 1994.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 313–320, New York, NY, USA, 1993. ACM Press.

[GBF03] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[GKJ+05] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph.*, 24(3):991–999, 2005.

[GKS02] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms: a simple framework for adaptive simulation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290, New York, NY, USA, 2002. ACM Press.

[GM03] Laurent Grisoni and Damien Marchal. High performance generalized cylinders visualization. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 257, Washington, DC, USA, 2003. IEEE Computer Society.

[Gri99] L. Grisoni. *Elements de multirésolution en modélisation géometrique*. PhD thesis, Université de Bordeaux I, France, 1999.

[GRLM03] Naga K. Govindaraju, Stephane Redon, Ming C. Lin, and Dinesh Manocha. Cullide: interactive collision detection between complex models in large environments using graphics hardware. In *HWWS '03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 25–32, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[GSLF05] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.*, 24(3):973–981, 2005.

[GVSS00] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 95–102, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[GW95] A. Guy and B. Wyvill. Controlled blending for implicit surfaces. In *Proc. Implicit Surfaces'95, pp. 107-112*, 1995.

[Hab97] A. Habibi. *Modèles Physiques Supports de la Relation Mouvement-Forme-Image*. PhD thesis, Institut National Polytechnique de Grenoble, janvier 1997.

[HAC03] Samuel Hornus, Alexis Angelidis, and Marie-Paule Cani. Implicit modelling using subdivision-curves. *The Visual Computer*, 19(2-3):94–104, May 2003.

[Hah88] James K. Hahn. Realistic animation of rigid bodies. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 299–308, New York, NY, USA, 1988. ACM Press.

[HPH96] Dave Hutchinson, Martin Preston, and Terry Hewitt. Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 31–45, New York, NY, USA, 1996. Springer-Verlag New York, Inc.

[HSB05] S. Hahmann, B. Sauvage, and G.-P. Bonneau. Area preserving deformation of multiresolution curves. *Comput. Aided Geom. Des.*, 22(4):349–367, 2005.

[Hub95] Philip M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995.

[JFS95] Charles E. Jacobs, Adam Finkelstein, and David H. Salesin. Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 277–286, New York, NY, USA, 1995. ACM Press.

[JJ03] J.S.M. Vergeest J. Jansson. Combining deformable and rigid body mechanics simulation. The Visual Computer Journal, 2003.

[Jou95] A. Joukhadar. Dynamic modeling system for robotics applications. In *Rapport de recherche INRIA n. 2543*, 1995.

[JP02] Doug L. James and Dinesh K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 582–585, New York, NY, USA, 2002. ACM Press.

[JP03] Doug L. James and Dinesh K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph.*, 22(1):47–82, 2003.

[JP04] Doug L. James and Dinesh K. Pai. Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. Graph.*, 23(3):393–398, 2004.

[JTT01] P. Jiménez, F. Thomas, and C. Torras. 3D Collision Detection: A Survey. *Computers and Graphics*, 25(2):269–285, April 2001.

[KEP05] Danny M. Kaufman, Timothy Edmunds, and Dinesh K. Pai. Fast frictional dynamics for rigid bodies. *ACM Trans. Graph.*, 24(3):946–956, 2005.

[KHM+98] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, /1998.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[LF04] Julien Lenoir and Sylvère Fonteneau. Mixing deformable and rigid-body mechanics simulation. In *Computer Graphics International*, pages 327–334, Hersonissos, Crete - Greece, june 16-19 2004.

[LGM+04] Julien Lenoir, Laurent Grisoni, Philippe Meseure, Yannick Rémion, and Christophe Chaillou. Smooth constraints for spline variational modeling. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 58–64, New York, NY, USA, 2004. ACM Press.

[LGMC05] Julien Lenoir, Laurent Grisoni, Philippe Meseure, and Christophe Chaillou. Adaptive resolution of 1d mechanical b-spline. Dunedin - New Zealand, 29 november-2 december 2005.

[LH91] David Laur and Pat Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 285–288, New York, NY, USA, 1991. ACM Press.

[LM87] Tom Lyche and Knut Morken. Knot removal for parametric b-spline curves and surfaces. *Comput. Aided Geom. Des.*, 4(3):217–230, 1987.

[LS00] Michael Lee and Hanan Samet. Navigating through triangle meshes implemented as linear quadtrees. *ACM Trans. Graph.*, 19(2):79–121, 2000.

[MAC04] D. Marchal, F. Aubert, and C. Chaillou. Collision between deformable objects using fast-marching on tetrahedral models. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–129, New York, NY, USA, 2004. ACM Press.

[MACOF+03] J. Behr M. Alexa, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Computing and rendering point set surfaces. *IEEE Trans. on Vis. and Comp. Graphics*, 9(1):3–15, 2003.

[Mal88] S. Mallat. *Multiresolution and wavelets.* PhD thesis, University of Pennsylvania, 1988.

[Man03] S. Mann. A geometric b-spline over the triangular domain. Master Thesis, University of Waterloo, 2003.

[Man05] Dinesh Manocha. Quick-cullide: Fast inter- and intra-object collision culling using graphics hardware. In *VR '05: Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 59–66, 319, Washington, DC, USA, 2005. IEEE Computer Society.

[Mes02] P. Meseure. Animation basée sur la physique pour les environnements interactifs temps-réel. Habilitation à diriger des recherches, Université de Lille 1, 2002.

[MHTG05] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, 2005.

[Mir00] Brian Mirtich. Timewarp rigid body simulation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 193–200, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[MKN+04] M. MÜller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, New York, NY, USA, 2004. ACM Press.

[MLFC04] P. Meseure, J. Lenoir, S. Fonteneau, and C. Chaillou. Generalized god-objects : a paradigm for interacting with physically-based virtual worlds. Computer Animation and Social Agents (CASA), Geneva, July, 2004.

[MTG04] Matthias Muller, Matthias Teschner, and Markus Gross. Physically-based simulation of objects represented by surface meshes. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 26–33, Washington, DC, USA, 2004. IEEE Computer Society.

[MW88] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animationr3. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1988. ACM Press.

[NS00] H.W. Nienhuys and A. F. Van Der Stappen. Combining finite element deformation with cutting for surgery simulation. In *Eurographice'00 Proc. (short papers), p. 43-51*, 2000.

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *ACM TOG (Proc. SIGGRAPH 2003), 22(3):463-470*, 2003.

[OD01a] C. O'Sullivan and J. Dingliana. Collisions and perception. ACM Transactions on Graphics. Vol. 20, No. 3. July, 2001.

[OD01b] Carol O'Sullivan and John Dingliana. Collisions and perception. *ACM Trans. Graph.*, 20(3):151–168, 2001.

[OHM+04] C. O'Sullivan, S. Howlett, Y. Morvan, R. McDonnell, and K. O'Conor. Perceptually adaptive graphics. Eurographics 2004, State of the Art reports. September, 2004.

[OL05] Miguel A. Otaduy and Ming C. Lin. Stable and responsive six-degree-of-freedom haptic manipulation using implicit integration. Proc. of the World Haptics Conference, pp. 247-256. Pisa, Italy, 2005.

[Pai02] D. Pai. Strands: Interactive simulation of thin solids using cosserat models. Computer Graphics Forum, 21(3):347–352, 2002. Proceedings of Eurographics'02., 2002.

[PDA00] Guillaume Picinbono, Herve Delingette, and Nicholas Ayache. Real-time large displacement elasticity for surgery simulation: Non-linear tensor-mass model. In *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 643–652, London, UK, 2000. Springer-Verlag.

[PKA+05] Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. Meshless animation of fracturing solids. *ACM Trans. Graph.*, 24(3):957–964, 2005.

[PPG04] Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. Quasi-rigid objects in contact. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–119, New York, NY, USA, 2004. ACM Press.

[Pra85] H. Prautzsch. Letter to the editor. *Computer Aided Geometric Design*, 2(4):329, 1985.

[Pro95] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In Wayne A. Davis and Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Human-Computer Communications Society, 1995.

[PT95] L. Piegl and W. Tiller. *The NURBS book*. Springer Verlag, 1995.

[Red04] S. Redon. *Continuous Collision Detection for Rigid and Articulated Bodies*. ACM SIGGRAPH Course notes, 2004.

[RGF+04] Laks Raghupathi, Laurent Grisoni, François Faure, Damien Marchall, Marie-Paule Cani, and Christophe Chaillou. An intestine surgery simulator: Real-time collision processing and visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 10(6):708–718, Nov/Dec 2004.

[RGL05] Stephane Redon, Nico Galoppo, and Ming C. Lin. Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3), 2005.

[RKC01] S. Redon, A. Kheddar, and S. Coquillart. Contact: In -between motions for collision detection. In *IEEE International Workshop on Robot and Automation*, 2001.

[RKC02a] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. *Computer Graphics Forum (EUROGRAPHICS Proc.)*, 21(3), 2002.

[RKC02b] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Gauss' least constraints principle and rigid body simulations. In *ICRA, IEEE International Conference on Robotics and Automation*, pages 517–522, 2002.

[RNG99] Y. Rémion, J. Nourrit, and D. Gillard. Dynamic animation of spline like objects. Proc. of WSCG'99, pp. 426-432,, 1999.

[SCF+04] Thomas W. Sederberg, David L. Cardon, G. Thomas Finnigan, Nicholas S. North, Jianmin Zheng, and Tom Lyche. T-spline simplification and local refinement. *ACM Trans. Graph.*, 23(3):276–283, 2004.

[Sch46] I. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly Journal of Applied Mathematics*, 4:45–99, 1946.

[Sch81] L. L. Schumaker. *Spline Functions: Basic Theory*. John Willey & Sons, New York, 1981.

[SDS96] E. J. Stollnitz, T. D. Derose, and D. H. Salesin. *Wavelets for computer graphics: theory and applications*. Computer Graphics and Geometric Modeling. Morgan Kaufmann, San Fransisco, CA, 1996.

[She99] A. Sherstyuk. *Convolution Surfaces in Computer Graphics*. PhD thesis, Monash University, Australia, 1999.

[SJP05] Le-Jeng Shiue, Ian Jones, and Jörg Peters. A realtime gpu subdivision kernel. *ACM Trans. Graph.*, 24(3):1010–1015, 2005.

[SP95] V. V. Savchenko and A. A. Pasko. Collision detection for functionally defined deformable objects. In *EUROGRAPHICS Workshop on Implicit Surfaces*, pages 217–221, Grenoble, France 18-19 avril 1995.

[SS95] Peter Schröder and Wim Sweldens. Spherical wavelets: efficiently representing functions on the sphere. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 161–172, New York, NY, USA, 1995. ACM Press.

[Sta99] Jos Stam. Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[Swe95] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[SZBN03] Thomas W. Sederberg, Jianmin Zheng, Almaz Bakenov, and Ahmad Nasri. T-splines and t-nurccs. *ACM Trans. Graph.*, 22(3):477–484, 2003.

[TGMC03] Frederic Triquet, Laurent Grisoni, Philippe Meseure, and Christophe Chaillou. Realtime visualization of implicit objects with contact control. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 189–ff, New York, NY, USA, 2003. ACM Press.

[THMG04] Matthias Teschner, Bruno Heidelberger, Matthias Muller, and Markus Gross. A versatile and robust model for geometrically complex deformable solids. In *CGI '04: Proceedings of the Computer Graphics International (CGI'04)*, pages 312–319, Washington, DC, USA, 2004. IEEE Computer Society.

[TKH+05] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics forum*, 24(1):61–81, March 2005.

[VSK95] O. Okunev V. Savchenko, A. Pasko and T. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Comp. Graph. Forum*, 14(4):181–188, 1995.

[Zac01] G. Zachmann. Optimizing the collision detection pipeline. In *First International Game Technology Conference (GTEC)*, 2001.

Annexe 1 : CV (English)

GRISONI Laurent, assistant professor
Professionnal address:
Université des Sciences et Technologies de Lille
LIFL - UMR CNRS 8022- Bâtiment M3
59655 Villeneuve d'Ascq Cédex - FRANCE
Phone number:
+33 320 434 492
Email :
laurent.grisoni@lifl.fr
URL :
http://www.lifl.fr/∼grisoni

Formation and employment

- September 2003-August 2005 : INRIA full-time researcher position
- 2000- ? : Assistant professor at Polytech'Lille engineering school, University of Lille 1.
- 1999-2000 : full time lecturer at University of Bordeaux I
- 1996-1999 : PhD thesis, defended on December,15th 1999, mention "très honorable", of University of Bordeaux I, prepared in LaBRI laboratory, advisor Christophe Schlick. Thésis title : Elements of multiresolution in geometric modeling.
- 1995-1996 :
  - ENSEIRB engineering school, computer science major. Option: computer graphics.
  - Computer science research Master (french DEA) of University of Bordeaux I, mention "Bien". Title: Multiresolution Curve and Surfaces (advisor Christophe Schlick).

Teaching activities from 2000 to 2005:

- Sept. 2003-Aout 2005 : INRIA full-time research position
- University years 2003-2004 and 2004-2005 : master course on advanced computer graphics, University of Lille 1.

- 2000-2003 :
  - Assistant professor at Polytech'Lille engineering school.
  - In charge of pedagogic content from September 2001 to August 2003 of computer science option, for 3rd year student, IMA departement.
- 2002-2003 : 214h of teaching (computer architecture, computer graphics, database, programmation, algorithmic)
- 2001-2002 : 340h (computer architecture, computer graphics, network, programmation, data compression, image analysis)
- 2000-2001 : 250h (computer architecture, computer graphics, database, programmation, algorithmic)

Research (PhD and Master thesis) advising

- PhD thesis of Julien Lenoir (defended on November 23rd, 2004), university of Lille 1, deformable 1D model for real-time physical simulation: the main targeted application of this work is surgical thread. This thesis proposed a spline deformable model, associated to a set of constraints solved using Lagrange multipliers (among which, handling of slipping point constraint, for surgical suture). This model is also derived in an adaptive version that provides real-time knot tightening at arbitrary position on the thread, and cutting at arbitrary locations.
- PhD thesis of Jérémie Dequidt (defense planned on December, 9th, 2005), started in September 2003, University of Lille 1: Autonomous objects in real-time physical simulation: this thesis defines the basis of our research activity dealing with multi-model objects in physical simulation. The work proposed a novel, flexible, architecture for physical simulation: each mechanical system is encapsulated within an autonomous agent. In order to point out the fact that such approach provides a software context in which objets can be easily associated to quite sophisticated features, a complete deformable model is proposed, that uses adaptivity both for collision detection, and mechanical simulation. This model is associated to some performance control box, that provides the model the ability to adapt its computation time to some predefined level, under some tolerance.
- PhD thesis of Adrien Theetten, (co-advised between INRIA Futurs and French CEA), started in December 2004, real-time animation of stiff deformable 1D models : this work aims at, through study of real-time simulation of deformation of significant section electrical wires, to propose mechanically consistent model for stiff deformable 1D model. The main targeted application, for our research team, is to propose consistent surgical thread animation model.
- PhD thesis of Radu Vatavu (cotutelle University Lille 1-University of Suceava, Romania), started in September 2004, free-hand interfaces for work in 3D environments: this thesis studies tools for real-time human gesture acquisition and interpretation for 3D work command. My (slight) involvement in this advising is due to my experience of splines and wavelet-based multiresolution, that potentially provide efficient mathematical tool for feature extraction within hand movement.

- Master thesis of Jérémie Dequidt, Collision detection between autonomous physical objects, University of Lille 1, July 2002 : this master thesis initiated the phd work of that student, and proposed first steps of physical simulation architecture, that provides mechanical systems with enough flexilibity on the collision detection aspects.
- Master thesis of Wenjing Lu, meshless for real-time simulation, and dynamic cutting operation, July 2005, University of Lille 1 (the student is now preparing PhD at IEMN).
- Master thesis of Sylvain Gaeremynck, Component-based architecture for collaborative 3D work, July 2003, University of Lille 1 (this student is currently preparing a Phd in our research team): this work, along with Jérémie Dequidt master thesis, was the first attempt to provide architecture for collaborative physical simulation
- Master thesis of Mr. Raboisson, subdivision-based squelettons for implicit objects deformations, July 2000, University of Bordeaux I: this master thesis aimed at studying practical advantages of using subdivision schemes (either for curves or surfaces) on squeletton for implicit objects.
- Master thesis of Ireneusz Tobor, extensions of the Lifting Scheme for splines, July 1999, University of Bordeaux I (the student then prepared a PhD on rendering surfaces using surfels, with C. Schlick as advisor, and defended in December, 2002): this master thesis studied extensions for the lifting scheme (a theorem providing flexible tools for biorthogonal wavelet constructions) and proposed simple biorthogonal decomposition for spline multiresolution editing.
- Master thesis of Guillaume Jourdain, applications of the Lifting Scheme for image analysis, University of Bordeaux I, July 1997.

Scientific collaboration activities

- We have been part, in 2002-2003, to some project (French INRIA ARC), in collaboration with IRCAD insitute (institute on research on intestine cancer, strasbourg, France) and EVASION inria project, led by Marie-Paule Cani. This project aimed at providing a real-time simulation of human intestine system, along with mesentery (tissue linking intestine to remainder of body). The main challenge task of this was that such a deformable model is highly self-colliding. Our involvement in that project was mainly in the second year of that project, where we took the lead of activities (for our research team). It resulted, from a contribution point of view, in an efficient visualization algorithm for generalized cylinder [9, 2]. From software point of view, this project produced a real-time simulator for intestine and mesentery system, based on the SPORE library, lead by Philippe Meseure. This was also our first attempt of combination of two simulation systems (including different initial simulation frequencies) into a single one, which significantly helped us understanding all practical problems that can arise from such situation.
- We are currently involved in an open-source, real-time simulation plateforme project. The SOFA project is the result of a collaboration between the CIMIT Simulation groupe (MIT/Harvard Hospital, leader Stephane Cotin), and INRIA projects EVASION (leader Marie-Paule Cani, INRIA Rhônes-Alpes) and EPIDAURE (leader Nicolas Ayache, Sophia-antipolis). This project was born when it appeared the CIMIT and ALCOVE project,

separately, had started quite the same work: propose a framework flexible enough so that people would reuse algorithmic blocks, and change, for a given object, collision representation, visual and mechanic representation, as independently as possible. We chose to join effort, and after EVASION and EPIDAURE joined us in this work, INRIA supported this project by providing 2 engineers, for 2 years, for this project. The 1.0 version of the SOFA plateform should be made public soon (hopefully in January, during MMVR conference). It provides multi-threaded support for simulation algorithmic (not only for drawing and/or haptic control), proper handling of group (for collision and constraints), and software structure that provides enough flexibility and genericity for people to consider functional blocks as exchangeable, and semantically start to build their own simulator efficiently.

- We initiated (both from scientific and administrative point of view) the PhD co-advising of Radu Vatavu. Christophe Chaillou and myself had had the occasion, several times, to visit the university of Suceava for teaching purposes. Professor Pentiuc came three times in Lille, as a visiting professor, several months each time.
- We collaborate with Claude Andriot, from CEA LIST laboratory (Fontenay-Aux-Roses), on the co-advising of Adrien Theetten, on the subject of thick deformable 1D objects.
- We are in charge, for ALCOVE project, of European project proposals :
  - We participated to the redaction of the initial project ODYSSEUS, Eureka European project. This project is lead by IRCAD. It involves Storz (germany), France Telecom R&D, and Simsurgery (Norway) as industrial partners; INRIA project EPIDAURE, EVASION and ALCOVE as academic research partners. It aims at providing, at a commercially exploitable level, collaborative tools for pre-surgical operation step, and also involves (mainly prospective) simulation aspects.
  - We have been the ALCOVE contact in 2003-2004 for Network of Excellence proposals Virtual Cloning (led by Georges-Pierre Bonneau, INRIA Rhônes-Alpes) and SURGETICA ; in the first project, we were in charge of collaborative 3D work WP redaction.
  - We proposed, as a "prime", a FET-Open proposal in 2004, of STREP financial format. The project CoDesign aimed at providing tools for 3D collaborative work on architectural modelling. The initial proposal was refused by EU. The involved partners of this proposal were UCL (London, UK, contact Anthony Steed), EVASION project (INRIA Rhônes-Alpes, contact Marie-Paule Cani), France Télécom R&D (Lannion, France, contact Dominique Pavy), Lucid Group (Liege, Belgique, contact Pierre Leclerq), école des mines de Nantes (contact Cédric Dumas), University of Suceava (Romania, contact Stefan-Gheorghe Pentiuc). This project is still on work, for further financial support.
  - Two proposals are currently under review process, for the September 2005 european project call for the ALCOVE project: USE (STREP format) and C-IT (IP format) projects. Both are submitted targeting strategic objective 2.5.9 of the FP6: "collaborative working environments". The first proposal, the USE project, is lead by ZGDV (Michael Schnaider), and involves 8 partners, including Bosch and Daimler-Chrysler. The second project (C-IT) is lead by COAT (Basel, Switzerland) involves 37 partners, among which Volvo, Siemens, Microsoft, France Telecom R&D, Daimler Chrysler, CEA, Philips.

Other scientific activities

- Member since 2003 (re-elected in 2004) of Eurographics French chapter board.
- Reviewer for SCCG'01, Eurographics 2003, Eurographics 2005, MMM'04, Shape Modeling international 2004, Journal of Computer Aided Design, Journal of Graphic Tools.
- Expert for evaluation of two project submissions to french national "ACI Masse de Données" grants during 2002-2003, and one project submission to French ARA MDMSA grant in 2005.
- Organisation of 12th national workshop on Animation and Simulation (incl. proceedings edition) (http://www.lifl.fr/gtas05)
- Substitute Member since 2003 of the University of Calais commision in computer science for researcher/lecturer hiring (french "CSE").
- Correspondant for LIFL lab. of INRIA electronic documentation center.
- Co-chair, with Alexander Pasko (Univ. Hosei, Japan) of the computer graphics session, free software meeting (LSM), Bordeaux (France), July 2001.
- Part of three PhD defense jury : Benjamin Schmitt (University of Bordeaux I, December 1st, 2002), Patrick Reuter (University of Bordeaux I, December 2003), Julien Lenoir (co-advisor, Lille I, November 2004).

Annexe 2: Full publication list

Journals

- [1] J. Dequidt, D. Marchal and L. Grisoni, **Time Critical animation of deformable solids**, (special issue, Proc. of CASA'05 conference), J*ournal of Computer Animation and Virtual Worlds* (J. Wiley & Sons ed.), 16(3-4), pp. 177-187, July 2005.
- [2] L. Raghupathi, L. Grisoni, F. Faure, D. Marchal, M.-P. Cani, C. Chaillou, **An Intestine Surgery Simulator: Real-Time Collision Processing and Visualization**, *IEEE Trans. On Visualization And Comp. Graphics*, 10(6), pp. 708-718, 2004.
- [3] J. Dequidt, L. Grisoni, P. Meseure, C. Chaillou, **Detection de Collision entre objets rigides convexes autonomes** (in french), *revue francaise de CFAO et d'informatique graphique*, 18, pp. 183-195, 2004.
- [4] L. Grisoni, C. Blanc, C. Schlick, **Hermitian B-splines**, *Computer Graphics Forum*, 18(4), pp. 237-248, december 1999.

International Conferences papers

- [5] J. Lenoir, P. Meseure, L. Grisoni and C. Chaillou, **Adaptive resolution of 1-D mechanical B-spline**, to appear in *Proc. of ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, 29 november-2 december, Dunedin, New Zealand, 2005.
- [6] J. Dequidt, D. Marchal and L. Grisoni, **Time Critical animation of deformable solids**, to appear in *Proc. of International Conference on Computer Animation and Social Agents (CASA)*, October 17-19, Hong-Kong, 2005 (also published in Journal of Computer Animation and Virtual Worlds).
- [7] J. Lenoir, P. Meseure, L. Grisoni and C. Chaillou, **A Suture Model for Surgical Simulation**, *Proc. of International Symposium on Medical Simulation*, pp. 105-113, Cambridge, Massachusetts (USA), june 2004.
- [8] J. Lenoir, L. Grisoni, P. Meseure, Y. Remion and C. Chaillou, **Smooth constraints for spline variational modeling**, *Proc. of ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, pp. 58-64, Singapore, june 2004.
- [9] L. Grisoni, D. Marchal, **High performance Generalized Cylinders Visualization** , *Shape Modeling'03* (ACM Siggraph, Eurographics, and IEEE sponsored) proceedings, pp. 257-263, Aizu (Japan), july 2003.

- [10] F. Triquet, L. Grisoni, P. Meseure, C. Chaillou, **Realtime Visualization of Implicit Objects with Contact Control** , *Proc. ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, Melbourne (Australia), february 2003.
- [11] J. Lenoir, P. Meseure, L. Grisoni, C. Chaillou, **Surgical Thread Simulation**, *MS4CMS'02 Proceedings* (ESAIM ed.), 12, pp.102-107, INRIA Rocquencourt, november 2002.
- [12] L. Grisoni, S. Degrande, C. Chaillou, E. Ferley, M.-P. Cani, J.-D. Gascuel, **SpinCAS: a step toward virtual collaborative sculpting** , *VRIC'02 Proceedings*, Laval (France), June 2002.
- [13] I. Tobor, C. Schlick, L. Grisoni, **Rendering by surfels**, *Graphicon 2000 Proceedings*, Moscow (Russia), August 2000.
- [14] L. Grisoni, C. Schlick, **Multiresolution representation of implicit objects**, *Implicit Surface'98 Proceedings*, (ACM Siggraph sponsored), pp 1-10, Seattle (USA), 1998.

Posters and short papers

- [15] J. Dequidt, L. Grisoni and C. Chaillou, **Collaborative interactive physical simulation**, technical sketch, to appear in *Proc. of ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, 29 november-2 december, Dunedin, New Zealand, 2005.
- [16] I. Tobor, L. Grisoni, C. Schlick, **Extensions of the Lifting Scheme for Splines**, *Curves and Surfaces'99*, Saint-Malo (France) Juillet 1999.

Technical reports

- [17] J. Dequidt, L. Grisoni, C. Chaillou, **Asynchronous Interactive Physical Simulation**, technical report no. RR-5338, INRIA Futurs, October, 2004.
- [18] L. Grisoni, B. Crespin, C. Schlick, **Multiresolution Implicit Representation of 3D Objects**, technical report no. 1217-99, LaBRI, Univ. Bordeaux I, 1999.
- [19] L. Grisoni, C. Schlick, C. Blanc **HB-splines: an interesting subset of B-splines for interactive CAD applications** technical report no. 1193-97, LaBRI, Univ. Bordeaux I, 1997.
- [20] L. Grisoni, C. Schlick, C. Blanc **An Hermitian Approach for Multiresolution Splines**, technical report no. 1192-97, LaBRI, Univ. Bordeaux I, 1997.

Other communications

- [21] A. Theetten, L. Grisoni, C. Andriot, C. Chaillou, **Simulation temps réel de câbles quasi-rigides**, *Proc. AFIG 2005*, 2005.
- [22] L. Grisoni, J. Dequidt, C. Chaillou, **Loose synchronization for real-time physical simulation**, invited talk, *1st ICT regional workshop on Virtual Reality*, Seoul (South Korea), january 2005.
- [23] L. Grisoni, J. Dequidt, C. Chaillou, **Virtual Reality applications and Frame-Rate control**, *Proc. of DAS'04* (IEEE Romania sponsored), may 2004.
- [24] J. Lenoir, P. Meseure, L. Grisoni and C. Chaillou **Simulation Physique de fils**. *9ème rencontre du groupe de travail animation et simulation*, 13-14 juin 2002, Bordeaux, France.

- [25] L. Grisoni, S. Degrande, C. Chaillou, **Technical issues for collaborative virtual sculpting**, *DAS'02 proceedings*, Suceava (Romania), May 2002.
- [26] J. Dequidt, L. Grisoni, P. Meseure, C. Chaillou, **Détection de collisions entre objets rigides convexes autonomes**, *AFIG'2002 Proceedings*, december 2002.
- [27] I. Tobor, P. Reuter, L. Grisoni, C. Schlick, **Visualisation par surfels** (in French). *AFIG'2000 Proceedings*, 2000.
- [28] L. Grisoni, C. Schlick, **L'approche hermitienne pour les splines a multiresolution** (in French). *AFIG'97 Proceedings*, 1997.

Annexe 3: Relevant publications for the research statement

We include in this document the publications we consider relevant for the addressed research Outline. All of those publications, including their companion videos, are available at the URL http://www.lifl.fr/∼grisoni .

- [TGMC03] F. Triquet, L. Grisoni, P. Meseure, C. Chaillou, **Realtime Visualization of Implicit Objects with Contact Control** , *Proc. ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, Melbourne (Australia), february 2003.
- [LGM+04] J. Lenoir, L. Grisoni, P. Meseure, Y. Remion and C. Chaillou, **Smooth constraints for spline variational modeling**, *Proc. of ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, pp. 58-64, Singapore, june 2004.
- [GM03] L. Grisoni, D. Marchal, **High performance Generalized Cylinders Visualization** , *Shape Modeling'03* (ACM Siggraph, Eurographics, and IEEE sponsored) proceedings, pp. 257-263, Aizu (Japan), july 2003.
- [LGMC05] J. Lenoir, L. Grisoni, P. Meseure, and C. Chaillou, **Adaptive resolution of 1-D mechanical B-spline**, to appear in *Proc. of ACM International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (Graphite)*, 29 november-2 december, Dunedin, New Zealand, 2005.
- [DMG05] J. Dequidt, D. Marchal and L. Grisoni, **Time Critical animation of deformable solids**, to appear in *Proc. of International Conference on Computer Animation and Social Agents (CASA)*, October 17-19, Hong-Kong, 2005 (also published in Journal of Computer Animation and Virtual Worlds).
- [DGC04] J. Dequidt, L. Grisoni, C. Chaillou, **Asynchronous Interactive Physical Simulation**, technical report no. RR-5338, INRIA Futurs, October, 2004.

Realtime Visualization of Implicit Objects with Contact Control

Frederic Triquet Laurent Grisoni Philippe Meseure Christophe Chaillou
LIFL (University Lille 1), UPRESA CNRS 8022, 59655 Villeneuve d'Ascq, France
[triquet|grisoni|meseure|chaillou]@lifl.fr

Abstract

This article presents an extension of classical marching cubes for implicit objects associated with blending control. In the framework of classical blend graph[Guy and Wyvill 1995], a technique for handling blending is presented, based on mostly boolean operations, and hence allowing efficient implementation. What is more, the resulting tesselation is topologically closer to the theoretical topology of the mathematical object considered than classical marching cubes tesselations. We also propose an optimization, that we call **partial time stamping**, that significantly diminishes the amount of redundant field values evaluation during marching cubes tesselation, and has the advantage of being compatible with blending control. This technique can be seen as the extension of an optimization technique described in [Triquet et al. 2001], that was not designed for blending control. Our technique is finally compared to classical marching cubes implementations, as a reference. In this final section it is shown that our technique permits high performance visualization of implicits objects combined with blending control.

1 Introduction

Implicit objects appeared to be very useful in several computer graphics fields, including for example soft object modeling[Wyvill et al. 1986; Bloomental et al. 1997; Bloomenthal 1995], and skinning animated models, especially for physically-based animation [Gascuel 1993; Cani 1998]. Such a modeling tool provides nice properties, including automatic blending, compact representation, and no topological constraints. Yet, some special cases suffer from those properties. Among them, blending is a property that has sometimes to be controlled. Figure 1 illustrates this problem. This object is an example of an implicit skinning: body skeleton, using implicit skin, provides a soft approximation of the resulting object. However, the blending between the arm and the hip is not desirable.

As a result, it appears useful to be able to control such a blending property. For that purpose, several techniques exist. Most of them rely on the notion of group between primitives: primitives from a given group can blend together, whereas no blending is allowed between primitives from different groups. Some techniques can also be combined with blending control: in order to simulate the natural behavior of soft and deformable objects touching each other, local deformations due to collisions may be added.

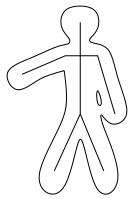


Figure 1: Typical blending problem

For real-time tesselation of implicit objects, two main classes of algorithms can be found: seed-based algorithms[Witkin and Heckbert 1994; Heckbert 1998; de Figueiredo et al. 1992; Desbrun et al. 1995] and space partitioning techniques[Wyvill et al. 1986; Lorensen and Cline 1987; Bloomenthal 1988; Bloomenthal 1994]. Those two classes of methods directly provide triangles used by graphics hardware. Seed-based techniques often handle topology using differential analysis techniques, and hence involve quite heavy computations, which make them difficult to use for high-performance tesselation. Beside, when handling dynamic implicit objects (i.e. objects composed of animated primitives), topological changes within the theoretical isosurfaces make the tesselation problem even more complicated. On the other hand, marching cubes based techniques are simpler to implement, and thanks to systematic treatments of space (at least in the basic definition of the algorithm [Lorensen and Cline 1987]), do not need any particular structuration of the object.

This paper proposes a tesselation technique combined with blending control that does not add any topological inconsistencies in regard to the theoretical result. The algorithm is presented, as well as the proposition of an adaptation of marching cubes optimizations previously published in [Triquet et al. 2001] to the problem of blending control, introducing what we called *partial time-stamping*.

The paper is organized as follows: the next section presents a rapid overview of previous techniques, both for blending control and deformation around contact areas. We also discuss them in the framework of real-time visualization. This section also points out the problems that occur for those techniques. Section 3 presents important points of a previously published paper dealing with high speed marching cubes. Starting from the problems raised in 2, we then propose our method in section 4, and show some experimental results in section 5.

2 Previous works

This section describes some previously published techniques that were proposed for controlling the blending between soft objects, as well as deforming the contact area, in order to simulate contact between deformable objects. We first enumerate techniques for those two problems, and then, discuss some technical issues about them in regard to marching cubes.

2.1 Blending control

Brian and Geoff Wyvill first pointed out the importance of blending control [Wyvill and Wyvill 1989]. They suggested to create primitive groups, within which blending is allowed, and with no interaction between primitives from different groups.

The idea consisting in distributing primitives among disjoined groups, works well for objects that have well-distinguished parts. Yet, in some special cases, this technique becomes quite involving: refer to the character of figure 1. It cannot be correctly represented in this way: his arms, legs and trunk have to belong to the same group in order to obtain correct blending of the surfaces at the shoulders and the hips, and as a consequence, arms and legs still blend together. Nevertheless, even if some group subdivision tricks could be used in this special case, the problem in the general case remains.

Wyvill's idea has been further improved in [Opalach and Maddock 1993; Guy and Wyvill 1995], where an adjacency graph describes the object structure. Each vertex of the graph stands for a primitive (that, by definition here, cannot be self-intersecting) and the edges represent a blend relation between primitives. In [Desbrun et al. 1995], the calculation of the potential at a given point P takes benefit of this structure by keeping only the maximal contribution of the non-blending involved skeletons. This contribution is calculated using classical summing. The four following steps sum the technique up [Desbrun et al. 1995]:

1. compute all the field contributions at point P,
2. select the field f_i that is preponderant,
3. add the maximal contribution from groups of skeleton that blend together and are all neighbors to S_i in the graph,
4. return the resulting value.

2.2 Local deformations

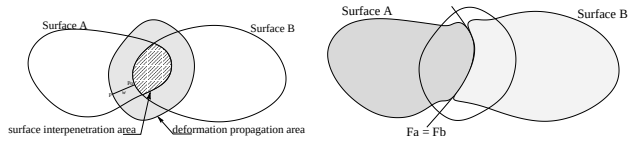
Cani-Gascuel[Gascuel 1993] describes a technique that controls blending between implicit primitives, and also simulates the contact between soft objects, by creating local deformations near the contact area.

The deformation of a surface a by another surface b is obtained by modifying the field function as follows:

$$G_{a,b}(x,y,z) = F_a(x,y,z) - F_b(x,y,z) + isovalue \qquad (1)$$

where F_a (respectively F_b) is the field function associated with surface a (resp. b) and G_a the one associated with the new deformed surface a (deformed by b). The idea to use a negative field function in order to compress such surfaces was formerly proposed by Blinn in [Blinn 1982].

When an inter-penetration is detected, each surface is deformed according to all the concerned objects. Within the inter-penetration zone, the contact area is determined by taking equation 1 and the symmetric expression (swapping a and b). As a result, the points lying on the contact area satisfy $F_a = F_b$ (see figure 2b). Around this inter-penetration



(a) Initial surfaces (b) Deformed surfaces

Figure 2: Contact and deformations between two implicit surfaces

zone, a given quantity is added to the field values. This quantity depends on the thickness w of the zone where deformations are propagated, a given parameter α characterizing the size of the created bulge and the distance between the point we consider and the inter-penetration zone. As shown on figure 3, the typical shape of this function (defined on $[0, w]$) keeps C^1 continuity at the exterior limit of the deformation area (see figure 2b).

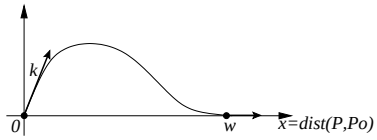


Figure 3: Shape of function used for field value deformation in [Gascuel 1993].

Opalach and Cani[Opalach and Cani 1997] simplify the problem: they build their deformation term using the field values associated with the surface causing the deformation (for example, surface B in figure 2). The shape of this function is shown in figure 4. In the deformation zone, the field

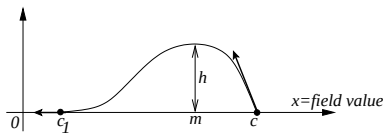


Figure 4: Shape of function D used in Equation 2 [Opalach and Cani 1997]

function for surface A becomes:

$$F_{A_2}(p) = F_A(p) + D(F_B(p)) \qquad (2)$$

where function $deform$, defined on $[c_1, c]$ satisfies the following properties:

$deform(c) = 0$	surface C^0 continuity
$deform'(c) = -1$	deformation around contact area
$deform(c_1) = 0$	C^0 continuity of the surface
$deform'(c_1) = 0$	C^1 continuity of the surface at the border of deformed zone

2.3 Discussion

Some special tesselation technique has to be derived when handling blending control and contact deformation. In other words, controlling the blending of implicit surfaces closely depends on the display scheme. The previously described methods for blend control give good results when used in a ray-tracing context or with a seed-based tesselation. However, they do not work well with a marching cubes algorithm. As a result, a specific visualization method is required.

For the tesselation computation, a problem occurs during the combination of marching cubes with classical deformation techniques. In order to illustrate this problem, let us focus on the results we get when applying, for example, Desbrun's field value computation[Desbrun et al. 1995] to the marching cubes algorithm.

Figure 5 shows the real implicit surface that should be approximated with triangles and the field values at the vertices of the grid. These field values result in the tesselation shown in figure 6a, which is slightly different to the one that should be built (figure 6b).

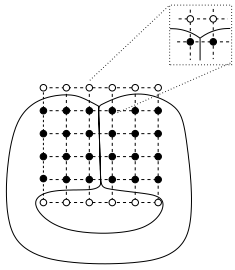
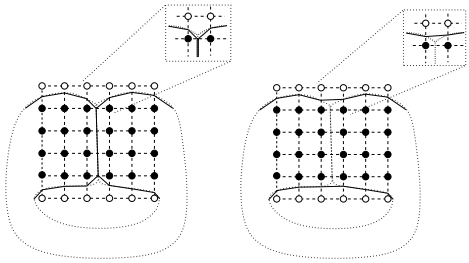


Figure 5: Initial configuration. The exact isosurface is drawn, as well as the marching cube grid. Black points are marked as being inside the surface whereas white points are considered as being outside the surface)



(a) (b)

Figure 6: Comparison between ideal tesselation (a) and constructed one (b) using classical marching cube

The tesselation we build is incorrect within the cubes containing both surfaces. For the considered configuration of those cubes, it is not possible to obtain a correct tesselation. Indeed, since we remember the field value of the preponderant primitive, the nodes lying on the left (respectively right) half of the grid correspond to the left-hand (resp. right-hand) primitive (figure 5). Thus, the two lower nodes of the figure 6 are considered to be *inside* the surface, while the two upper nodes are considered to be *outside*. Therefore, there is no reason for the marching cubes algorithm to build an intersection point along the lower edge (see figure 6b). It appears natural, at this step, to allow the marching cubes to do several partial tesselations whithin such a cube and to superimpose them in the final rendering..

When using a graph for controlling primitives blending, one generally tesselates separately each group of blending primitives. Such a method ensures topological consistency in regard to contact. Yet, for reasons described in [Guy and Wyvill 1995], it is not possible, in the general case, to keep the idea of grouping primitives in order to handle the case of self-colliding objects in such a manner. As a result, one has either to subdivide groups, or (and this is our choice here) to consider that the notion of group does not hold. Each primitive is considered separately. Such an assumption does not solve the classical problem of self-colliding object, since we consider that no primitive can be self-colliding. This is always true in the case of discrete skeletons. The generic problem is encountered when controlling the blending of a self-colliding, one piece object (and which topology has to be preserved, as for the character on figure 1). See for example the snake presented in figure 8, made using blobs. A "by group" decomposition, as previously presented, is not possible. On the other hand, seeing this snake as the union of small, simple, not self-intersecting nor self-colliding (i.e. blobs), primitives, makes the use of blending graph possible, and handling of auto-colliding snake, conceivable.

The notion of primitive group does not hold under the assumptions defined above. The question hence remains to know how to use the partial tesselation idea presented by Wyvill. In the framework of marching cubes tesselation, the notion of "group" within which all the primitives blend, can actually be found, locally to a tesselation cube. Indeed, for a cube several primitives have a non-null contribution to the global potential field. Among this list, it is theoretically possible, using the blend graph, to retrieve some sub-lists of primitives such that all the primitives of a sub-list blend together, and that no blending relation between primitives of different sub-lists does exist. For the considered cube, those local sub-lists are the groups that can be used for partial tesselation. Tesselating a cube does not involve a single classical tesselation, but the amount of sub-lists for a given cube defines the number of tesselations that are necessary for this cube.

3 Fast marching cubes

An implementation, in the "always blending" case, has been presented in detail in [Triquet et al. 2001]. In this article, a collection of algorithmic techniques is used in order to reach small tesselation times even for quite complex iso-surfaces (about 200 primitives in 25 ms on a high-end consumer PC). We sum up here important parts of this article in regard to the problem addressed (i.e. adaptation to implicit objects combined with blending control), so that the present paper is somewhat self-contained, and that the arguments that makes the proposed algorithm derivation important, quite natural.

In this algorithm[Triquet et al. 2001], the sources of optimizations to take into account are:

- First, space partitioning, well-known for improving ray-tracing[Cazals et al. 1995], collision detection, radiosity[Cohen-Or et al. 1998], is used for primitives sorting: the tesselation space is partitionned using simple cubes. For non-ambiguous description, we will, in the remainder of this paper, call *cube* the space partitioning unit used during marching cubes tesselation,

and *voxel* the space partitioning unit used for primitive grouping. For each voxel, a list of primitives is built, made of primitives which influence area intersects the considered voxel. This ensures that, for any point of a voxel, **including** the points of the frontier, the primitives that have non-null influence on the point are all among the voxel primitive list (although some of this list might have null influence). See figure 7 for an example of list construction. It can be seen that for a given voxel, it might happen that the overall potential evaluation does not involve all the primitives belonging to the voxel list.

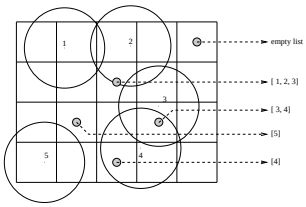


Figure 7: Example of voxel partitioning, combined with 5 primitives. Each primitive belongs to all the voxel lists its influence area intersects.

- For each cube, an unsigned integer value is stored, representing the frame number at which the cube has been treated. This number, called *time-stamp* identifies the current tesselation. At the end of a given tesselation, all the processed cubes have the same time-stamp value. Checking if a cube has already been treated during the current tesselation is equivalent to comparing the current time-stamp with the cube time-stamp. Such marking permits fast tesselation. It is likely that temporal coherency might be used for further optimizations. Yet, since our main framework is that of highly deformable objects (see section 5), such coherency had only but few chances to achieve real improvement (indeed, even small deformations induce new tesselation), and hence, was not the priority of our work.

- In order to combine both multipart objects handling and the minimization of treated cubes amount, a preprocessing of the global primitive list stacks a cube laying on the isosurface for each primitive. Time-stamping ensures that no cube can be stacked twice.

- A cube is then popped from the stack. This cube is used as a seed, and cubes are propagated on the isosurface, in quite a classical manner[Bloomenthal 1995]. Time-stamping value checking prevents multiple treatments.

- We store the vertices field values of each treated cube. Using time-stamping, it is quite easy to avoid redundant field evaluation, using previously calculated values: for a given cube, considering the neighbors the cube shares vertices with, and their time-stamp, one can easily decide whether the neighbor field values can be reused for the shared vertices: if the neighbor time-stamp value equals the current time-stamp value, then the field values are consistent for the current tesselation.

The next section describes in a more detailled manner our technique of tesselation, based on marching cubes. We describe a technique for extraction of primitive sub-lists for a cube, using only bitwise logical operations, and therefore allowing optimal implementation. We also describe some specific optimizations.

4 Implementation of the Blending Control

The framework we are interested in here is the tesselation of an object composed of simple, non-self-colliding primitives (see section 2 for discussion). Apart from the primitives, a blend graph is built, defining the way the primitives blend all together.

4.1 Global tesselation algorithm overview

The algorithm we propose here is derived from the one described in section 3. The main steps of this algorithm are:

- the same notion of *cube* and *voxel* is present here, respectively for tesselation cube, and space partitioning;

- the tesselation of a given frame F implies (same as section 3, but more precise, and with specific notations):

  - calculation of the new current time-stamp t_F, which only involves an integer addition.

  - Let us call N the total number of primitives P, all stored in a list $L = \{P_0, \cdots, P_{N-1}\}$. For each primitive P_i, a cube of the isosurface is calculated, using simple propagation in a particularized direction, and is stacked for further *treatment*.

  - Each cube of the stack is used as an original seed for tesselation: until the stack is empty, a cube c is popped off the stack, *treated*, then marked as treated using the time-stamp value t_F, and the cubes among c neighbors that have older time-stamp value are stacked for treatment. See section 4.3 for details about the time-stamp value, and its precise use. Due to previous stack creation from primitives list, it could happen that a cube might be treated twice: A simple time-stamp value check prevents this.

- The *treatment* of a cube c for a frame F implies:

  - Just like for technique described in section 3, a list L_c of primitives is built. L_c can be rewritten as the set $\{P_{\sigma(0,c)}, \cdots, P_{\sigma(n_c-1,c)}\}$, where σ stands for a function that maps the n_c primitives that have non-null influence on c to their original indices in L. See section 4.2 for further details.

  - L_c is partitioned into sub-lists $L_{c,0}, \cdots, L_{c,\alpha[c]-1}$. Those sub-lists are such that they represent, locally, the classical blend graph: Let define $P_a \in L_{c,i}$ and $P_b \in L_{c,j}$. Then P_a and P_b blend together if and only if $i = j$ (i.e. do not blend if $i \neq j$). See section 4.2 for details about the technique used for partitioning L_c. Note that when the blending relation is not transitive within a cube, then blend is not an equivalence relation, and no partitioning is possible. The solution we adopt in this case is to duplicate "pivot" primitives between non blending groups: for example, if P_i blends with P_j, P_j with P_k, but P_k does

not blend with Pi, then there will be two groups, $\{P_i, P_j\}$ and $\{P_j, P_k\}$.

– c is tesselated $\alpha[c]$ times in a marching cube like manner, the i-th tesselation corresponding to the primitive sub-list $L_{c,i}$. Each tesselation involves field evaluation using deformation techniques described in section 2.2.

Let us examine the critical case of the snake (Figure 8): here, the blending relation are composed of the groups $\{P_i, P_{i+1}\}$. In other words, the matrix representing the blending graph is a band matrix, with band width equal to 3, all elements of it being equal to 1. The duplication process we use entails that all primitives appear in two groups. Several groups will hence be computed, causing as many tesselations of the cube. However, such a case tends to be really rare since it only occurs when several (non blending) primitives lie within the same cube.

4.2 Efficient primitives sorting

We describe here a technique for manipulating the blend graph and creating the sub-lists needed by the global tesselation algorithm. The blend graph is represented by a two dimensional binary matrix M such that $M(i,j) = 1$ if P_i and P_j blend together, and 0 otherwise. In our C++ implementation, M is stored as a matrix of *long long unsigned int*, which has the drawback to constrain the number of primitives manipulated to be a multiple of the size of this datatype, but presents the important advantage to be very compact in regard to further manipulation, as we will see in this section. As a result, and without loss of generality, we can consider that the line $v(i)$ of matrix M is composed of an unsigned binary value, which j-th bit expresses if P_i blends with P_j or not.

Our framework here is that of a cube c for which we have built the list L_c. The very first step of the manipulation is the correspondance to be done between the list and a binary word (that we will abusively note using the same notation) where the i-th bit value is 1 if $P_i \in L_c$, and 0 otherwise. It is very practical to see all the involved lists as binary words of that kind, as, for example, testing if $P_i \in L_c$ can be written in a C-like syntax using bitwise operations, $L_c\&(1 << i)$. Merging two primitive lists A and B (e.g. for building the list L_C from the voxel lists) is simply evaluating $A|B$. In the result, P_i is present if the i-th bit equals 1. On most common nowadays architecture, we thus take benefit of merging lists of 64 primitives using one single bitwise logical operation. Using this idea, the operations we manipulate for extracting the sub-lists $L_{c,0}, \cdots, L_{c,\alpha[c]-1}$ from L_c are all part of a simple loop:

1. $\alpha[c] = 0$;
2. while L_c is not null:
   - calculate i as the first non-null bit in L_c /*this can be done using iterative shifting combined with AND masking*/;
   - retrieve $v(i)$ from M;
   - $L_{c,\alpha[c]} = L_c\&v(i)$;
   - $L_c = L_c \wedge L_{c,\alpha[c]}$ /*the sub-list $L_{c,\alpha[c]}$ is withdrawn from current list L_c. This, because in our special case we have $L_{c,\alpha[c]} \subset L_c$, that corresponds to a bitwise logical XOR operation */;
   - $\alpha[c] = \alpha[c] + 1$;

For the sake of simplicity, we did not include in this algorithm the special case of non-transitive blending relation within a cube list. Yet, it is quite simple to use bitwise logical operations for detecting the primitives that have to be shared by groups, and use this information for creating correct sub-lists. Such an algorithm description presents the immediate advantage to be both of a very high description level, and easily implemented in a very simple and efficient code, fully optimized by most common compilers.

4.3 Partial time stamping

Classical time stamping[Jevans et al. 1988], within the framework of blending controlled implicit object tesselation, is not fully usable: Its main goal was to prevent redundant potential evaluation by assigning a tesselation time to each cube. When treating a cube, the neighbor concerned by a given shared face is first checked, and, if its time-stamp value corresponds to the current value, then this means that the potential value stored can be used. When controlling blending, several tesselations per cube are possible, and as a result it is useless to know if the cube has the correct time-stamp, since several potentials would be present. Therefore, the question would remain, to know which value is correct for the primitives we consider.

Yet, such a philosophy can be reused under some slightly stronger assumptions: Within an implicit object having non-blending primitives, a high percentage of cubes does not involve any number of sub-lists greater than 1, i.e. do not have non-blending local groups. As a result, those cubes, under the condition that the neighbor does not need several treatments either, could take benefit of time-stamping anyway.

This is why we defined what we called *partial time-stamping*, which is a binary word made of the concatenation of classical time-stamp value, and a boolean information, expressing if the $\alpha[c]$ value, calculated by the algorithm presented in section 4.2, equals 1 or not. This simply allows for determining if we can re-use the previously calculated field values or not. The value stored as time-stamp per cube c, instead of value t_F for classical time-stamping, is now (in C-like syntax) $(t_F << 1)|(\alpha[c] == 1)$.

5 Experimental results and measures

This work takes place in a more complex software environment, being devoted to physical-based simulation. In figure 8 is presented a surface made of 100 primitives where the group technique cannot be applied, as the object is "continuously" defined, and self colliding.

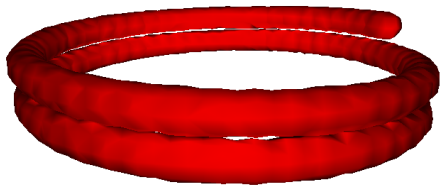


Figure 8: One hundred primitives with blending control

Figure 9 displays the wire version of the tesselation of 3 interacting blobs. One can see that the deformations applied to the surfaces are realistic, and that the topology of

the result matches the intuitive topology of the theoretical isosurface.

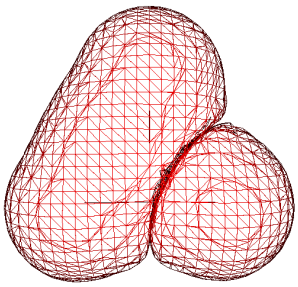


Figure 9: Wireframe display of 3 primitives (blobs)

Figure 10 shows the time required to tesselate a surface, depending on the number of primitives. Those measures have been performed on various versions of the object shown on figure 8, being deformed periodically like a spring along its axis.

A comparison is made in figure 10 between the tesselation times required by different implementation techniques: Our initial tesselator (without blending control), the technique presented in this paper (with blending control), and Bloomenthal's implementation[Bloomenthal 1995] (which, in its initial formulation, cannot consider blending control). Bloomenthal's code, since it has been written to be easily understood, was not very optimized. However, the space partitioning technique can speed up a lot the tesselation process, as shown in figure 10.

Note that in those measures, only the tesselation time is given, not the time needed for visualizing the resulting object (which is graphics hardware dependant, and independant from the algorithm described in this paper).

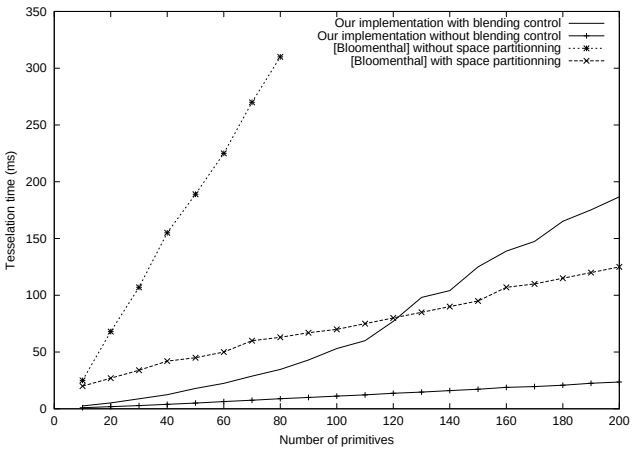


Figure 10: Tesselation time using several techniques

The tesselation method presented in this paper is used in a simulator of laparoscopic surgery which, for accurate simulation reasons, has to be an application with high framerate performances. In this simulator, intestines are modeled using 165 primitives. Its tesselation, made of about 40000 triangles, is entirely constructed at each frame. Due to the high deformability of the model, and the context of physical simulation that introduces small, yet existing, oscillations, no time coherency is used. The overall application, using our tesselation technique combined with real-time physical manipulation of the intestines, based on physical simulation, runs approximatively at 7 images per second on a Pentium IV at 1.7GHz using a *Quadro II* graphic card (although we can reach 23 images per second in the same context when desactivating the blending control). Figure 11 shows a laparoscopic tool manipulating these intestines. As it can be seen, our technique allows us to control the blending and the deformations of the surfaces built around complex skeletons.

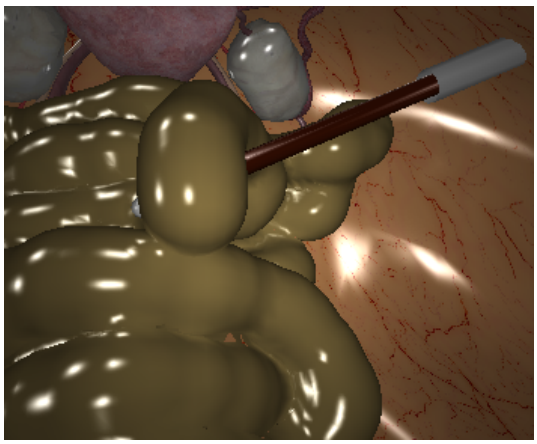


Figure 11: Manipulating implicit intestines in laparoscopic surgery simulation

6 Conclusion and future work

This paper presented a framework for high-performance tesselation of implicit objects, that provides blending control. The proposed tesselation technique does not introduce further topological inconsistencies, in regard to classical ambiguities introduced by marching cubes. The blend graph proposed implementation allows for treatment using only bitwise logical operations, hence simply enables optimal results. We also presented an optimization technique called partial time stamping that significantly diminishes redundant field value evaluation.

An interesting question, above all in regard to the optimizations proposed for for marching cubes, would be to study the applicability of this technique to other tesselation algorithms, such as marching tetrahedrons [Bloomental et al. 1997]. Another direction of research would also be to see in which measure the work presented here can be adapted to the case of continuous skeletons, such as convolution surfaces based on subdivision skeletons [Cani and Hornus 2001]. In this new framework, the assumption stating that the primitives used are not self-colliding no longer holds, and we think that multiresolution-based tesselation techniques [Grisoni et al. 1999] would be a way to take this problem into account.

References

Blinn, J. F. 1982. A generalization of algebraic surface drawing. *ACM Transactions on Graphics 1*, 3 (July), 235–256.

Bloomental, J., Bajaj, C., Blinn, J., Cani-Gascuel, M., Rockwood, A., Wyvill, B., and Wyvill, G. 1997. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc.

Bloomenthal, J. 1988. Polygonisation of implicit surfaces. *Computer Aided Geometric Design 5*, 341–355.

Bloomenthal, J. 1994. An implicit surface polygonizer. *Graphics Gems IV*.

Bloomenthal, J. 1995. Bulge elimination in implicit surface blends. In *Implicit Surfaces'95*, 7–20. Proceedings of the first international workshop on Implicit Surfaces.

Cani, M.-P., and Hornus, S. 2001. Subdivision curve primitives: a new solution for interactive implicit modeling. In *Shape Modelling International*.

Cani, M.-P. 1998. Layered models with implicit surfaces. In *Graphics Interface (GI'98) Proceedings*. Invited paper, published under the name Marie-Paule Cani-Gascuel.

Cazals, F., Puech, C., and Drettakis, G. 1995. Filtering, clustering, and hierarchy construction: a new solution for ray tracing complex scenes. In *EUROGRAPHICS'95 Proc.*, 371–382.

Cohen-Or, D., Fibich, G., Halperin, D., and Zadicario, E. 1998. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. *Computer Graphics Forum 17*, 3. Eurographics'98 Proc.

de Figueiredo, L., Gomes, J., Terzopoulos, D., and Velho, L. 1992. Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface'92*, 250–257.

Desbrun, M., Tsingos, N., and Gascuel, M.-P. 1995. Adaptive sampling of implicit surfaces for interactive modelling and animation. In *Implicit Surfaces'95*, 171–186. Proceedings of the first international workshop on Implicit Surfaces.

Gascuel, M.-P. 1993. An implicit formulation for precise contact modelling between flexible solids. *Computer Graphics 27*, 313–320.

Grisoni, L., Crespin, B., and Schlick, C. 1999. Multiresolution Implicit Representation of 3D Objects. Technical report, University of Bordeaux I, http://www.lifl.fr/~grisoni.

Guy, A., and Wyvill, B. 1995. Controlled blending for implicit surfaces. In *Implicit Surfaces'95*, 107–112. Proceedings of the first international workshop on Implicit Surfaces.

Heckbert, P. 1998. Fast surface particle repulsion. *Report CMU-CS-97-130, April 1997. Appeared in SIGGRAPH 98 course notes*.

Jevans, D., Wyvill, B., and Wyvill, G. 1988. Speeding Up 3D Animation For Simulation. *Proc. SCS Conference*, 94–100. Proc. MAPCON IV (Multi and Array Processors).

Lorensen, W., and Cline, H. 1987. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4 (July), 163–169. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

Opalach, A., and Cani, M.-P. 1997. Local deformations for animation of implicit surfaces. *SCCG'97 (Bratislava, Slovakia)* (jun). http://w3imagis.imag.fr/Membres/Marie-Paule.Cani/mouImplicite.html.

Opalach, A., and Maddock, S. 1993. Implicit surfaces: Appearance, Blending and Consistency. In *Fourth Eurographics Workshop on Animation and Simulation*, 233–245.

Triquet, F., Meseure, P., and Chaillou, C. 2001. Fast polygonization of implicit surfaces. *WSCG'2001 (Plzen, Czech Republic) 2* (feb), 283–290. http://wscg.zcu.cz.

Witkin, A., and Heckbert, P. 1994. Using particles to sample and control implicit surfaces. *Computer Graphics* (July), 269–278. Proceedings of SIGGRAPH'94.

Wyvill, B., and Wyvill, G. 1989. Field functions for implicit surfaces. *The Visual Computer 5* (Dec.), 75–82.

Wyvill, G., McPheeters, C., and Wyvill, B. 1986. Data structure for soft objects. *The Visual Computer* (Aug.), 227–234.

Smooth constraints for spline variational modeling

Julien Lenoir* IRCICA-LIFL, Lille
Laurent Grisoni† IRCICA-LIFL, Lille
Philippe Meseure‡ SIC, Poitiers
Yannick Rémion§ LERI, Reims
Christophe Chaillou¶
ALCOVE Project, INRIA Futurs, IRCICA-LIFL,
University of Lille 1, FRANCE

Abstract

This article introduces a new class of constraints for spline variational modeling, which allows more flexible user specification, as a constrained point can "slide" along a spline curve. Such constraints can, for example, be used to preserve correct parameterization of the spline curve. The spline surface case is also studied. Efficient numerical schemes are discussed for real-time solving, as well as interactive visualization during the energy minimization process. Examples are shown, and numerical results discussed.

CR Categories: I.3.5 [COMPUTER GRAPHICS]: Computational Geometry and Object Modeling—Physically based modeling; I.3.5 [COMPUTER GRAPHICS]: Computational Geometry and Object Modeling—Splines; J.6 [COMPUTER-AIDED ENGINEERING]: Computer-aided design; I.6.5 [SIMULATION AND MODELING]: Model Development—Modeling methodologies; I.6.0 [SIMULATION AND MODELING]: Model Development—General

Keywords: variational modeling, smooth constraint, spline, physically based modeling, dynamic resolution

1 Introduction

Among the important properties often appreciated in geometric modeling, stands the ability for a given model to be easily edited. This property partially explains the popularity of models such as splines [Piegl and Tiller 1997] and subdivision surfaces [Zorin and Schroder 2000]. Yet, in some special cases, many degrees of freedom for a geometric model can be quite a drawback to design simple, natural shapes, for example. Such a manipulation can be quite long and difficult for the user, making the model not suitable for such a goal. In order to solve this problem, deformation tools [Coquillart 1990; Barr 1984] are often used in order to provide simple edition for models with many degrees of freedom. Some models also have multiresolution features [Forsey and Bartels 1988; Grisoni et al. 1999] that provide a control of the number of degrees of freedom used during manipulation. This way, the user only manipulates the model at the desired resolution, and can precisely control the influence of the modifications.

Another possible solution is to use variational modeling [Welch and Witkin 1992; Gortler and Cohen 1995; Witkin et al. 1987], which principle is to combine initial condition to be respected, and a physical solver that determines the shape fitting the conditions and minimizing some energy criterion (classically, the global curvature). Constraints can thus be defined dynamically on the model to help the designer during the manipulation. The most popular constraints are those which affect a specific point of a curve or a surface (for instance, a fixed point, tangent, normal, etc...). Other constraints show themselves useful for direct manipulation like constraining a curve or surface to pass through a specific point (in free space or on an object). This has been already proposed in the past for particular objects.

It can be useful to easily define some sliding points on a shape, i.e. ensure that some point of the shape is at a specified location in space, whatever the corresponding parameter value of the point. For example, some applications in direct edition process, by fixing enough sliding points on the curve, we can entirely define its shape by specifying a kind of path. Such a constraint ensures that parameterization will also be taken into account during the energy minimization step. Apart from the parameterization point discussed above, such a constraint would offer additional degrees of freedom to variational modeling. This article presents a method for handling such constraints.

We use Lagrange formalism to describe our technique, and introduce what we call *smooth constraints* [1]. Classically in variational modeling, a static resolution process is used during the energy minimization step. This ensures a proper system resolution, but presents the drawback that the user does not have any control during the resolution process. We propose another approach, that uses dynamic resolution, and allows the user both to interact with the model during energy minimization, and control the numerical evolution of the system. In order to allow an efficient solving, we propose a modification of the constraint equations.

The paper is organized as follows: Section 2 consists in an overview of the previous work followed by section 3 presenting the principle of variational modeling, including notations and general equations. The spline case is especially developed in a sub-section. The next two sections present the theory for smooth spline curve constraints and the equations modifications needed to achieve high performance real-time resolution. A set of smooth constraints for a spline curve is then exposed in a section 6. Section 7 generalizes the theory to spline surfaces. Finally, numerical results, and examples, are presented in section 8 before concluding.

*email: Julien.Lenoir@lifl.fr
†email: Laurent.Grisoni@lifl.fr
‡email: philippe.meseure@sic.sp2mi.univ-poitiers.fr
§email: yannick.remion@univ-reims.fr
¶email: Christophe.Chaillou@lifl.fr

[1] It is to note that the term 'smooth constraint' is used in [Robinson 2003], associated to a different meaning. In our context, 'smooth constraint' refers to constraints specified without explicit parameter value (e.g. spline curve that slides through static space point).

2 Previous Work

In a general way, variational modeling searches the ideal emplacement of a model by minimizing an energy criterion (for example, the thin plate energy employed by Pottmann et al. [2002] for an active contour application). This approach can be done statically [Witkin et al. 1987] or dynamically [Qin and Terzopoulos 1996]. The employed energy can be seen in term of constraints that can be solved by a physical simulation. The interest in variational modeling resides in the possible constraints that can be applied to a model to ease its manipulation by the end-user. But creating constraints dynamically can yield some problems like discontinuity or an unsolvable system due to the resolution process. In this perspective, Gleicher [1992] proposed among others a direct edition method for constraints definition.

Welch and Witkin [1992] characterize a surface with a spline formulation and define an energy criterion based on a simplified expression from Terzopoulos et al. [1987]. Then, they define geometric constraints, finite-dimensional constraints and also transfinite constraints dealing with the entire surface (global energy). The resolution process is static and such proposed constraints do not allow a specific point of one object to be locked on another object surface (global consideration of the surface with evolving localization).

Among convenient constraints, Witkin et al. [1987] proposed higher levels tools. One of them obliges an object surface to pass through a specific point without specifying the actual object point. Their object can be defined by different ways, set of discrete points, iso-surface of implicit functions. By this way, a specific point of an object can easily be locked on the surface of another object defined by its implicit form. This is a pioneering work and a fundamental solution to the smooth constraint problem. The solution proposed a very interesting form of constraint but it has the drawback that the object defined by its implicit form is known by its surface. Nevertheless, it is in no way a trivial purpose to define a curve by an implicit form because it is not a natural way to create a 1D model especially for a deformable model. Even the use of a distance map brings computation time issues. Moreover, the static process of resolution needs energy local minima and thus the object configuration can jump from a solution to another one discontinuously.

Qin and Terzopoulors [1996] proposed a dynamic NURBS to simulate a curve and adapted it for geometric modeling by exploiting force reaction to bring local deformation and gravity to help designers. By a tensor product, they extend their tool to surface. They enforce their constraint by the way of Lagrange multipliers and Baumgarte stabilization scheme. They use augmented Lagrange technique but constraints which are not localized on the model can not be simulated by such method combination. The dynamic formalism proposed by Qin is interesting because it offers a temporal coherence of the behavior and thus avoid jumps which occur in a static process. We propose to complete this dynamic formalism in order to take into account a new class of constraints.

3 Dynamic Variational Modeling

3.1 Generalities

Traditionally, variational modeling consists in considering an objective function to be minimized, along with some specific constraints [Welch and Witkin 1992]. The objective function is generally based on an energy expression of the curve, surface or volume.

All these considerations can be simulated thanks to the lagrangian theory, that aims at minimizing the energy of a system in a dynamic process. Moreover, it can be directly extended to take into account some specific constraints, as in the classic way of variational modeling, by use of the Lagrange multipliers. In order to animate a curve, we consider this object as a set of degrees of freedom introduced in a mechanical system. With these degrees of freedom, we explain the mechanical law thanks to the lagrangian equations [Lenoir et al. 2002; Rémion et al. 1999]:

$$\begin{cases} \forall i, \frac{d\frac{\partial K}{\partial \dot{q}_i}}{dt} + \frac{\partial K}{\partial q_i} = Q_i + L^T.\lambda \\ \phi(q_i, \dot{q}_i) = 0 \end{cases} \tag{1}$$

where K is the kinetic energy of the system, q_i the degrees of freedom, Q_i the power of the different potential forces. L is a matrix defined using the different constraints (ϕ) relatively to all degrees of freedom [Rémion 2000]. λ are the Lagrange multipliers, each of them is related to the force intensity needed to preserve an associated constraint.

Classically, equations (1) are derived into a linear system:

$$\begin{pmatrix} M_g & L^T \\ L & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A} \\ -\lambda \end{pmatrix} = \begin{pmatrix} \mathbf{B} \\ \mathbf{E} \end{pmatrix} \tag{2}$$

where M_g is the generalized mass matrix of the system ones considers (resulting from the kinetic energy term), L the constraints matrix, A the acceleration of the degrees of freedom, B a vector that sums the different contributions of all external and inertial forces and finally E a vector coding the intensity of the violation of the different constraints.

By this way, we can fix a constraint directly on the degrees of freedom or anywhere on the object (degrees of freedom are not necessarily on the object as in the BSpline case) or between different objects present on the same dynamic system (like two patches with a common edge).

3.2 Spline Variational Modeling

The chosen model is a spline geometry on which we apply the mechanical laws. For a 1D spline, the position of a point on the curve is defined as a function of s (its parametric abscissa):

$$P(s,t) = \sum_{i=1}^{n} q_i(t).b_i(s) \tag{3}$$

where q_i are the control points, n the number of control points, b_i the basis functions linked to the spline type, s the parametric abscissa of the point and t the time (for our test, we used Catmull-Rom spline, cubic uniform BSpline and non uniform BSpline).

To simulate a spline, we define the three coordinates q_i^α of each of its control points as the degrees of freedom[2] of the dynamic system which thus has a size of $3n$. After some computations, we obtain:

$$\begin{aligned} \forall i, \frac{d\frac{\partial K}{\partial \dot{q}_i^\alpha}}{dt} &= m.\sum_{j=1}^{n}\int_0^1 b_i(s).b_j(s).ds.\ddot{q}_j^\alpha \\ \forall i, \frac{\partial K}{\partial q_i^\alpha} &= 0 \end{aligned} \tag{4}$$

where m is the mass of the object. We thus obtain a diagonal block generalized mass matrix with the same block for each axis:

$$M_g = \begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{pmatrix} \tag{5}$$

[2] thus, their accelerations are noted $\ddot{q}_i^\alpha$, which is equivalent to $\frac{d^2 q_i^\alpha}{dt^2}$

thanks to the equations (4), we easily define the components of M by:

$$M_{ij} = m.\int_0^1 b_i(s).b_j(s).ds \tag{6}$$

What we desire here is both the global and local modification of the curve when the end user changes the configuration. We can obtain a local modification by a direct edition of a point but the global shape of the curve does not change. We need here an internal energy permitting the propagation of the local modification to the entire curve. Since, we are interested in doing variational modeling, gravity is not involved. Q_i represents the influence of some springs that are associated to the curve for the internal energy [Lenoir et al. 2002], and the forces due to the interaction with the user. Those equations can also add forces given by a scene collision detection or self-collision process when needed.

The constraints are taken into account via the Lagrange multipliers λ. They allow to constrain a system, by the way of its degrees of freedom. We can thus constrain directly the degrees of freedom or any point of the object (for a BSpline, the degrees of freedom are the control points which do not belong to the curve). For example we can fix coordinates of any point on the curve by applying:

$$FixedPointConstraint(s_0,t) = P(s_0,t) - A$$

which declines in three simpler constraints, one for each axis. By the same way, constraints for fixing one or two coordinates are possible.

According to existing literature, such constraints can only be relative to a specific point of the curve (or more basically on the degrees of freedom themselves). We are thus not able to consider a constraint onto the entire curve. With such constraints, it is not possible to constrain the curve to pass through a point in space without specifying which constant point of the curve should be placed there. The next section presents a solution to this problem.

4 Smooth spline constraints

We wish to give more flexibility to the constraints by making them non localized on the curve. So that the effective point where the constraint is applied depends on the dynamics itself.

One example, is the sliding point constraint, where some point of the object must be at a specific location, but this curve point is not always the same step after step, depending on the dynamics. In this example, we just have to fix a point of the curve, considering the fact that it is not always the same point that is fixed during the process stabilization. As a curve point is defined by a parametric abscissa s, the specific s parameter of the fixed point constraint must be dynamic and thus depends on time : $s(t)$.

Clearly, our constraint is a fixed point constraint based on a dynamic abscissa parameter. Such a constraint g is written:

$$g(q,\dot{q},t,s(t)) = P(s(t),t) - P_0 \tag{7}$$

This equation imposes that some point of the spline must be at the position P_0. But it is a dynamic system so, the $s(t)$ will change over time in order to find the right point of the curve that minimizes the energy of the constrained system.

A problem occurs: The g equation demands to the system to verify a condition but the system ensures that the constraint will be fulfilled at a stable state but not necessarily immediately. This introduces a numerical drift on the constraint over time. To take into account this feature , we use the equation of g to formulate a second order differential equation that gives us a damp solution with a critical damping [Rémion et al. 1999; Baumgarte 1972]. This leads to the constraint equation:

$$\ddot{g} + \frac{2}{\delta t}\dot{g} + \frac{1}{\delta t^2}g = 0 \tag{8}$$

where δt is the time step used during the simulation.

By considering the equation (7), we obtain the suitable constraint equation:

$$\frac{d^2P}{dt^2} + \frac{2}{\delta t}\frac{dP}{dt} + \frac{1}{\delta t^2}(P - P_0) = 0 \tag{9}$$

In the development of this equation (using expression of P given by equation (3), new terms appear that do not exist for a simple fixed point constraint:

$$\begin{aligned}
&\sum_{i=1}^{n}(\ddot{q}_i(t).b_i(s(t)) + q_i(t).b_i'(s(t)).\ddot{s}(t)) = \\
&\quad - \sum_{i=1}^{n}(2.\dot{q}_i(t).b_i'(s(t)).\dot{s}(t) + q_i(t).b_i''(s(t)).\dot{s}(t)^2) \\
&\quad - \frac{2}{\delta t}\sum_{i=1}^{n}(\dot{q}_i(t).b_i(s(t)) + q_i(t).b_i'(s(t)).\dot{s}(t)) \\
&\quad - \frac{1}{\delta t^2}(\sum_{i=1}^{n} q_i(t).b_i(s(t)) - P_0)
\end{aligned} \tag{10}$$

In fact, we can see that the constraint equation needs the dynamics of the s parameter. We thus have to consider it as an unknown of our system. It will be numerically integrated step by step like all other unknowns in order to determine its new position and velocity. This is a particular use of the lagrangian formulation because we just define a new unknown that is neither a degree of freedom nor a Lagrange multiplier. Because of this new unknown we have to find an additional equation in order to entirely define the system equation. This equation can give us an idea of the evolution of the s parameter or explain a constraint, linking s to the physical system.

If we consider a perfect constraint without friction, the lagrangian theory imposes that the virtual power of the strain due to this link must be equal to zero. In other words, the force generated by the Lagrange multipliers must not work in mechanical terms. For that, the Lagrange multipliers λ of this constraint must verify the equation:

$$\lambda.\frac{\partial g}{\partial s} = 0 \tag{11}$$

This theoretical framework is explained in more details in [Rémion 2003].

We obtain a global system:

$$\begin{pmatrix} M & 0 & 0 & 0 & Lx^T \\ 0 & M & 0 & 0 & Ly^T \\ 0 & 0 & M & 0 & Lz^T \\ 0 & 0 & 0 & 0 & Ls^T \\ Lx & Ly & Lz & Ls & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A^x} \\ \mathbf{A^y} \\ \mathbf{A^z} \\ \ddot{\mathbf{s}} \\ -\lambda \end{pmatrix} = \begin{pmatrix} \mathbf{B^x} \\ \mathbf{B^y} \\ \mathbf{B^z} \\ \mathbf{0} \\ \mathbf{E} \end{pmatrix} \tag{12}$$

the new matrix Ls describes the different constraints according to the new unknowns $\ddot{\mathbf{s}}$. In other words, it introduces the sliding point constraint into the system. Ls is composed by the terms of the constraint equations (10) where $\ddot{s}$ appears. For a single sliding point constraint, Ls is a (3×1) matrix composed by the column vector $\frac{\partial g}{\partial s}$.

5 Resolution

The equation system (12) is quite complex to solve because we do not have a direct relation that gives the new value of $\ddot{s}$. For an interactive application, such a resolution can degrade the performance and thus make the manipulation difficult because of the latency of the animation. Consequently, we choose to consider equation (11) in a different way.

We accept that the effective work of the force generated by the Lagrange multipliers is not null and we represent it as an error. This error is due to an incorrect value of s and is applied to correct the dynamics of this parameter. This approach gives us an equation slightly different from equation (11):

$$\varepsilon.\ddot{s} - \lambda.\frac{\partial g}{\partial s} = 0 \tag{13}$$

In order to stay close to the theoretical framework, we multiply the acceleration of s by a factor ε close to zero. This little modification of the equation gives this new system of equations:

$$\begin{pmatrix} M & 0 & 0 & 0 & Lx^T \\ 0 & M & 0 & 0 & Ly^T \\ 0 & 0 & M & 0 & Lz^T \\ 0 & 0 & 0 & \varepsilon & Ls^T \\ Lx & Ly & Lz & Ls & 0 \end{pmatrix} \begin{pmatrix} \mathbf{A^x} \\ \mathbf{A^y} \\ \mathbf{A^z} \\ \ddot{\mathbf{s}} \\ -\lambda \end{pmatrix} = \begin{pmatrix} \mathbf{B^x} \\ \mathbf{B^y} \\ \mathbf{B^z} \\ \mathbf{0} \\ \mathbf{E} \end{pmatrix}$$

In this system, the matrix L is defined by its components: $(LxLyLz)$. Ls is an extension of L which permits to explain constraints on the $\ddot{s}$ unknowns.

The equation (13) gives us a direct relation to find the value of the $\ddot{s}$ unknown and thus accelerates the resolution process.

We decide to solve the system by decomposing the acceleration in two parts, one of *tendency* and one of *correction*: $A = A_t + A_c$ [Rémion 2000]. The acceleration of tendency represents the acceleration without any constraint and the other acceleration is the correction due to the constraints. This leads us to this new equation system:

$$\begin{cases} M.A_t^x = B^x \\ M.A_t^y = B^y \\ M.A_t^z = B^z \\ M_g.A_c = L^T.\lambda \\ \varepsilon.\ddot{s} = Ls^T.\lambda \\ L.(A_t + A_c) + Ls.\ddot{s} = E \end{cases}$$

We replace the terms A_c and $\ddot{s}$ in the sixth equation by their expression respectively in the fourth equation and the fifth equation. These replacements yield an equation for λ:

$$\begin{cases} M.A_t^x = B^x \\ M.A_t^y = B^y \\ M.A_t^z = B^z \\ A_c = M_g^{-1}.L^T.\lambda \\ \varepsilon.\ddot{s} = Ls^T.\lambda \\ L.M_g^{-1}.L^T.\lambda + \frac{Ls.Ls^T}{\varepsilon}.\lambda = E - L.A_t \end{cases}$$

We called n the number of control points of the spline so we have $M(n \times n)$ and $M_g(3n \times 3n)$. c is the number of constraints in the system, and thus is the height of L and Ls. And c_g is the number of new unknowns in the system, and thus is the width of Ls.

M is both constant over time and symmetric band matrix (cf. equation (6)) due to the spline locality property, we thus pre-compute an LU decomposition where L and U are two band matrices too. The computation time of A_t is then linear (thus in $\mathcal{O}(n)$).

We also can pre-compute the inverse matrix of M and use it to compute $M_g^{-1}.L^T$ then the matrix $R = L.M_g^{-1}.L^T$, this computation is in $\mathcal{O}(c.n^2 + c^2.n)$. We complete R by the integration of the sliding constraints part: $\frac{Ls.Ls^T}{\varepsilon}$. This completion is done in $\mathcal{O}(c_g.c^2)$.

Finally, we solve the matrix equation $R.\lambda = E - L.A_t$ which takes a complexity of $\mathcal{O}(n.c + c)$ for the right term computation, $\mathcal{O}(c^2)$ for R decomposition and $\mathcal{O}(c^2)$ for the final resolution. This gives us the values of the Lagrange multipliers and permits us to compute the correction accelerations $A_c = M^{-1}.L^T.\lambda$ (resolution in $\mathcal{O}(n.c)$) and the new value of the supplementary unknowns: $\varepsilon.\ddot{s} = Ls^T.\lambda$ (resolution in $\mathcal{O}(c.cg)$).

The trick employed in (13) permits to resolve the system in $\mathcal{O}(c.n^2 + c^2.n + c_g.c^2)$.

For comparison, a direct resolution gives a complexity of $\mathcal{O}((n + c_g + c)^3)$. By applying the same technique of acceleration decomposition on the unmodified system (12), this yields three different accelerations by dissociating the c constraints into c_1 usual constraints and c_2 smooth constraints ($c = c_1 + c_2$) [Rémion 2003]. This decomposition allows a resolution in the same order of complexity as the above resolution, but takes more computation stages. So the theoretical framework would be more time consuming.

6 Set of Smooth Constraints

Thanks to this free variables technique, a running knot can be easily defined using the constraint equation:

$$g(q, \dot{q}, t, s(t), s_0) = P(s(t), t) - P(s_0, t)$$

Such a constraint defines knot on a curve and also gives us the possibility to manipulate it easily, and change its position on the curve.

In the same way, a double sliding point constraint can be defined on a spline by the equation:

$$g(q, \dot{q}, t, s_1(t), s_2(t)) = P(s_1(t), t) - P(s_2(t), t)$$

This last constraint equation allows to manipulate a knot with the desired piece of spline. This gives us much more freedom on the manipulation but requires a second unknown.

Another application of this technique, is the reusing of the new free variables in others constraints equations. For example, we can imagine a sliding point constraint representing a surgical thread during a suture. At the contact point, the curve tangent can be constrained perpendicularly to the organ surface. This can be done by a tangent sliding point constraint equation:

$$g(q, \dot{q}, t, s(t), s_0) = \frac{dP(s(t), t)}{ds} - T(s_0, t)$$

where $T(s_0, t)$ is the tangent vector wanted at the $s(t)$ abscissa parameter.

We can imagine by the same way, a constraint on the normal vector:

$$g(q, \dot{q}, t, s(t), s_0) = \frac{d^2P(s(t), t)}{ds^2} - C(s_0, t)$$

with $C(s_0, t)$ is the curvature vector wanted at the $s(t)$ abscissa parameter.

Many constraint equations can be created by this way and new experimentation must be explored. Furthermore, this technique is practicable in the 2D case.

7 Generalization to surface

This formulation can be easily generalized to a 2D object, we just have a constraint based on the degrees of freedom. In this case, we would have two free variables for defining a sliding constraint. In terms of internal energies, we can apply some springs [Provot 1995] or trying a continuous energy [Terzopoulos et al. 1987; Nocent and Rémion 2001]. For a 2D spline (of size $n \times n$), we have the expression:

$$P(u,v,t) = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij}(t).b_i(u).b_j(v)$$

Then, the K term in the lagrangian equation gives us a much more complex mass matrix than the 1D case:

$$\frac{\partial K}{\partial q_{ij}} = (m. \sum_{i_1=1}^{n} \sum_{j_1=1}^{n} \int_0^1 \int_0^1 b_i(u) b_{i_1}(u) b_j(v) b_{j_1}(v) du dv) \dot{q}_{i_1 j_1}$$

As a result, M is composed of terms:

$$M_{(i_1 j_1),(i_2 j_2)} = m. \int_0^1 \int_0^1 b_{i_1}(u).b_{i_2}(u).b_{j_1}(v).b_{j_2}(v).du.dv$$

by adopting the coding line/column, a point of index i is the $q_{(i/n)(i\%n)}$ point, it gives a more precise formulation:

$$M_{ij} = m. \int_0^1 \int_0^1 b_{i/n}(u).b_{j/n}(u).b_{i\%n}(v).b_{j\%n}(v).du.dv$$

Such a matrix has a $n^2 \times n^2$ size (for a patch of size $n \times n$) and a band structure thanks to the spline locality that localizes the interactions between the control points in the 2D structure. A point interacts with some neighbors that are on the same line or on the neighbors line. These lines are localized in the matrix structure closely to the current one. In conclusion, we obtain a band matrix much larger than the 1D case.

Algorithms such as *Cuthill Mckee* also permit to re-organize the structure of the matrix by giving a different numeration of the knot[3]. It may enhance the resolution process by giving a more interesting structure to the M matrix.

A sliding point constraint for a surface is defined by the equation:

$$g(u(t),v(t),t) = P(u(t),v(t),t) - A$$

this constraint needs two free variables u and v to define a single sliding constraint. The computation will be more fastidious than the 1D case but the process is very similar.

Such constraint permits to define a point on the tissue through which the surface may slide while the user pulls it by an extremity. In a manner similar to a tissue that would slide on a stake if a person pulls it by one of its corner.

In term of complexity, the mass matrix grows to a $n^2 \times n^2$ size and always has a band property but with a larger band width. This remark set apart, the complexity is the same as in the 1D case.

8 Results

All the tests have been performed on a Pentium IV, 2.4GHz and 512Mb.

We first show that our method is effective with this animation example: A shoelace is pulled from one extremity (figure 1). The first image represents the initial situation where the shoelace is correctly put on the shoe, the holes are modeled using white spheres. The second picture shows what happens when we just pull it slowly. The shoelace slides along the sliding point constraints. The last picture represents the state of the dynamic process later when the shoelace passes through the two first holes. This last picture brings up the interesting property that each constraint can be activated or deactivated dynamically and automatically when the constraint becomes outside the curve, and can no longer be valid. The apparent rigidity of the shoelace is due to the discretization we chose for the spline (we could have a better movement by taking a much more subdivided spline, at the cost of a longer computation). In this example, the dynamic resolution process takes about 12ms for each computation step.

The second result shows the advantage of this method in regard of the distribution of the energy along the spline (figure 2). On the picture, the user manipulates the curve thanks to a probe which can be linked to a control point (which one can see in the second and third images). The crosses symbolize the control points of the spline in order to show their distribution. The first image is the initial situation where we take a spline on which we fix the extremities. Starting from this, we either fix its middle point (second picture) or define it as a sliding point (third picture). The result shows that the manipulation changes the modeling of the spline but the fixed point can be seen as two splines with constraint on the joining point, which localize the manipulation on one of the two sub-splines and some modification on the neighbor sub-spline due to the links constraint. For a sliding constraint, the spline is considered as a unique spline that we ask to pass through a specific point. The sliding process enables the spline to distribute its energy onto the entirely curve by passing the sliding point. This allows the control points to be evenly distributed along the curve.

The next example shows a slipknot constraint (figure 3). With such a constraint, we can make a knot with a spline and manipulate the spline in order to clamp the knot or on the contrary make it wider.

A useful application of this proposition is the direct manipulation of the sliding point constraint location. By this way, the model is accessible by its control points and also by its sliding points. Thus we impose that the curve passes through a point whose position is interactively set by the end-user(figure 4).

In figure 5, snapshots of sliding point constraints simulations on 2D spline are shown. The first image shows a 7×7 patch with three sliding points and one fixed point. The resolution process takes about 20ms for each computation step. The second image shows another configuration with the same patch but with only one sliding point constraint. For this example, the computation time is about 14ms.

In the different pictures, a green sphere represents a fixed point constraint and a white sphere represents a sliding point constraint.

9 Conclusion and Prospect

This article proposed a dynamic approach for the variational modeling and shared a specific constraint on a point without specifying explicitly the parameter of this point. With this type of constraint, we are able to make a direct edition and then offer a new way of modeling a spline.

This technique can be easily used in animation and could give interesting results for simulating all sorts of threads, ropes, etc.

It would be interesting to adapt the resolution of the spline curve depending on the constraints the user applies. Using multiresolution splines [Finkelstein and Salesin 1994] combined with automatic decision criteria, would be a valuable tool. Such a work is currently in progress.

[3] Boost Graph Library, Cuthill McKee Ordering. http://www.boost.org/libs/graph/doc/cuthill_mckee_ordering.html

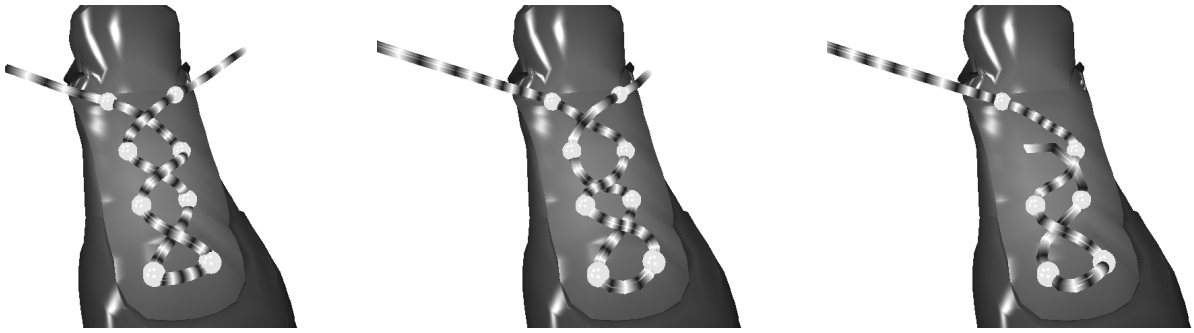


Figure 1: A shoelace sliding on some shoe hole (white spheres)

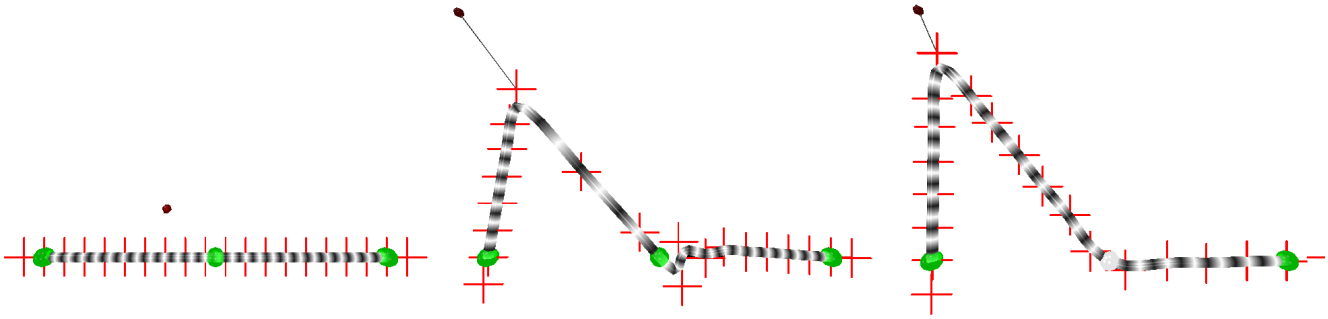


Figure 2: Left: Initial spline - Middle: Tension created with a fixed point
right: Correct re parameterization with a sliding point - (Control points are shown by the red crosses)

References

BARR, A. 1984. Global and local deformations of solid primitives. In *Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, 18, 3, ACM, 21–26.

BAUMGARTE, J. 1972. Stabilization of constraints and integrals of motion. *Computer Meth. Appl. Mech. Eng. 1*, 1–16.

COQUILLART, S. 1990. Extended free-form deformation: A sculturing tool for 3d geometric modeling. In *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, 24, 4, ACM, 187–193.

FINKELSTEIN, A., AND SALESIN, D. H. 1994. Multiresolution curves. In *Proceedings of ACM SIGGRAPH 94*, ACM Press/ACM SIGGRAPH, New York., Computer Graphics Proceedings, Annual Conference Series, ACM, 261–268.

FORSEY, D., AND BARTELS, R. 1988. Hierarchical b-spline refinement. In *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, 22, 4, ACM, 205–212.

GLEICHER, M. 1992. Integrating constraints and direct manipulation. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, ACM, 171–174.

GORTLER, S. J., AND COHEN, M. F. 1995. Hierarchical and variational geometric modeling with wavelets. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, ACM, 35–42.

GRISONI, L., BLANC, C., AND SCHLICK, C. 1999. Hermitian b-splines. *Computer Graphics Forum 18*, 4 (Dec.), 237–248.

LENOIR, J., MESEURE, P., GRISONI, L., AND CHAILLOU, C. 2002. Surgical thread simulation. In *Proceedings of Modelling and Simulation for Computer-aided Medecine and Surgery (MS4CMS)*, EDP Sciences, Rocquencourt, vol. 12, INRIA, 102–107.

NOCENT, O., AND RÉMION, Y. 2001. Continuous deformation energy for dynamic material splines subject to finite displacements. In *Proceedings of the Eurographic workshop on Computer Animation and Simulation*, Springer Verlag, Manchester (UK), 87–97.

PIEGL, L., AND TILLER, W. 1997. *The NURBS Book*, second ed. Springer-Verlag, New York.

POTTMANN, H., LEOPOLDSEDER, S., AND HOFER, M. 2002. Approximation with active b-spline curves and surfaces. In *Proceedings of Pacific Graphics*, 8–25.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics Interface'95 Conference*, 147–154.

QIN, H., AND TERZOPOULOS, D. 1996. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 85–96.

RÉMION, Y., NOURRIT, J., AND GILLARD, D. 1999. Dynamic animation of spline like objects. In *Proceedings of the WSCG'1999 Conference*, 426–432.

RÉMION, Y. 2000. Animation dynamique : moteur lagrangien généraliste et applications. *Habilitation à diriger des recherches, Université de Reims, Champagne-Ardenne.* (Dec.).

RÉMION, Y. 2003. Prise en compte de "contraintes à variables libres". Tech. Rep. 03-02-01, LERI, Feb.

ROBINSON, S. M. 2003. Variational conditions with smooth constraints: Structure and analysis. In *International Symposium on Mathematical Programming*, 245–265.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISHER, K. 1987. Elastically deformable models. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21, 4, ACM, 205–214.

WELCH, W., AND WITKIN, A. 1992. Variational surface modeling. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, 26, 2, ACM, 157–166.

WITKIN, A., FLEISCHER, K., AND BARR, A. 1987. Energy constraints on parameterized models. In *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21, 4, ACM, 225–232.

ZORIN, D., AND SCHRODER, P. 2000. Subdivision for modeling and animation. *ACM SIGGRAPH 2000, course notes 23* (July).

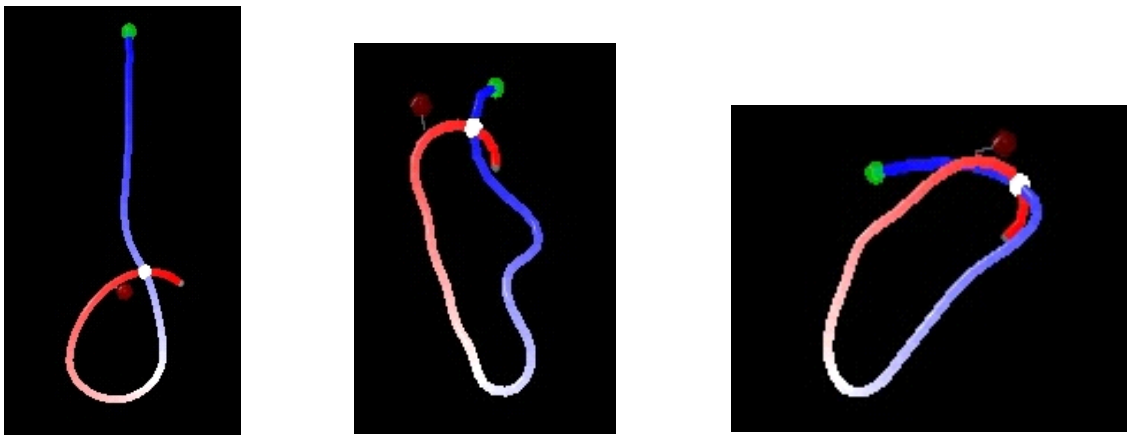


Figure 3: A rope for hang: a rope with a sliding point constraint linked to another point of the curve

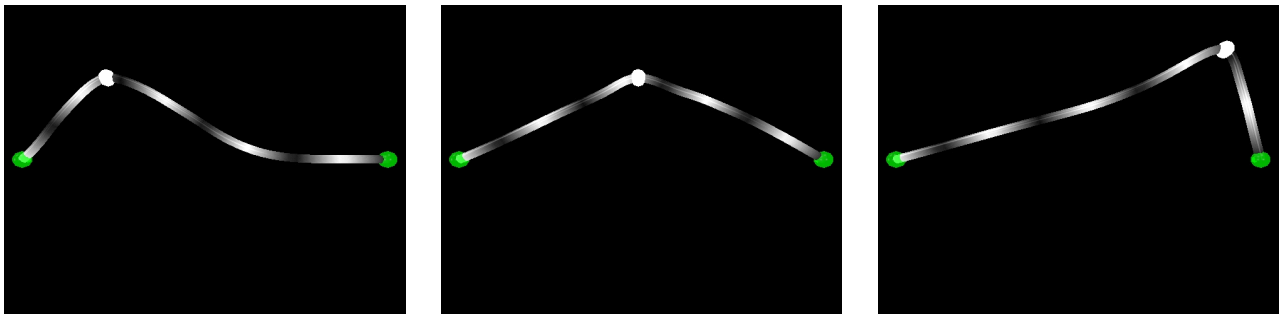


Figure 4: Direct Edition with an interactive sliding point constraint

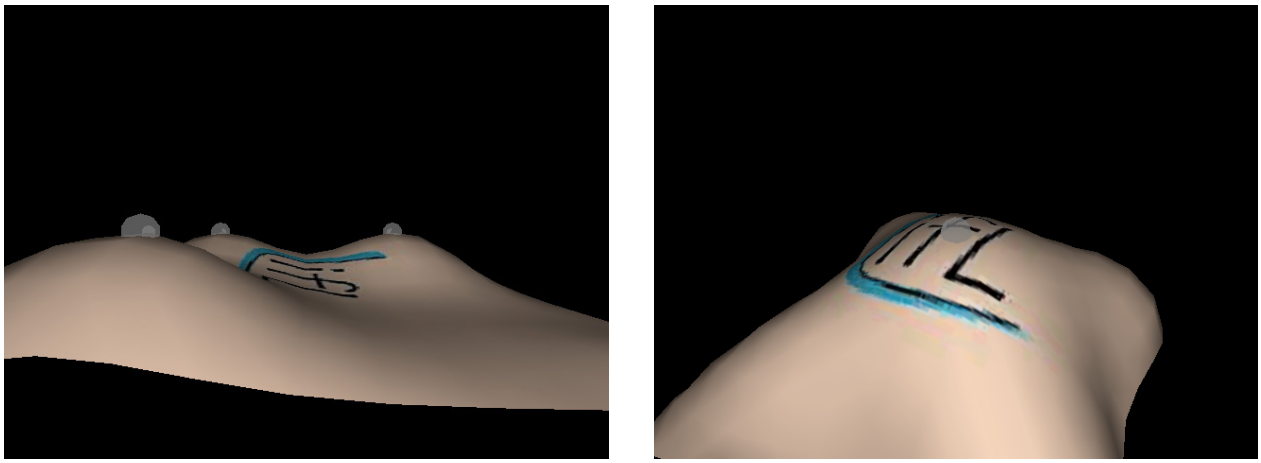


Figure 5: Application to 2D splines

High Performance generalized cylinders visualization

Laurent Grisoni Damien Marchal

LIFL, University of Lille 1, UPRESA CNRS 8022, Batiment M3
59655 Villeneuve d'Ascq Cedex, France
[Laurent.Grisoni|Damien.Marchal]@lifl.fr

Abstract

This paper studies the problem of highly deformable generalized cylinders real-time rendering. Some efficient schemes for high axis curvature detection are presented, as well as an incremental non-uniform sampling process. We also show how the recent 3D card "skinning" feature, classical in character animation, can be derived in order to allow for very high frame-rate when rendering such generalized cylinders. Finally, an algorithm is presented, that permits the object to dynamically adapt its display process, for guaranteed frame-rate purposes. This algorithm dynamically modifies the different sampling parameters in order to achieve optimal quality visualization for a given pre-imposed frame-rate.

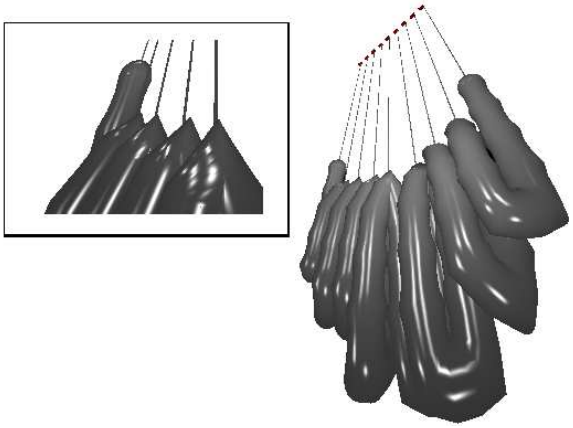


Figure 1. classical uniform tessellation of a simple generalized cylinder. Top-left: zoom on axis high-curvature area.

1. Introduction

Widely used in computer graphics modeling, generalized cylinders are now quite commonly used. Since their original definition [3], several declinations of the original model have been presented, and extensively studied [8, 9, 14, 19]. One major advantage of such a model is that a 3D shape is conceptually simplified into several 1d-curves, and the process of such a shape design, simplified to several 1d-function definition [8]. This allows for efficient design of topologically simple objects. One of the major problems often encountered with such models is that of conversion to polygonal approximation (this process is called *tessellation*), e.g. for further real-time visualization. It is well known that simple poly-line extrusion raises tessellation problems for high-curvature points (see Figure 1 for an example of it).

This comes from two main problems when handling high-curvature points on the cylinder axis: first, it may occur that the sharp angle is simply missed in the sampling. The second problem is the fact that tessellated sections can overlap, hence providing the user with self-intersecting mesh (see figure 2 for illustration of those two problems).

This paper presents three contributions to the problem of high-performance visualization of such objects: first, we present an incremental technique for non-uniform sampling of the cylinder axis. Second, we introduce another way of constructing the polygonal approximation used for fast visualization. Last, we present an algorithm that allows for the rendering process to adapt itself to a specific frame rate, which can be quite useful when designing graphically complex applications that might possibly be executed on a computer with limited rendering possibilities.

The paper is organized as follows: section 2 defines the notation used in this paper, as well as classical techniques for high-curvature detection along the axis, and generalized cylinder rendering. Section 3 presents our incremental solution for adaptive sampling of the axis. Section 4 shows how hardware-based skinning can be extended to a subclass of generalized cylinders, and significantly reduce

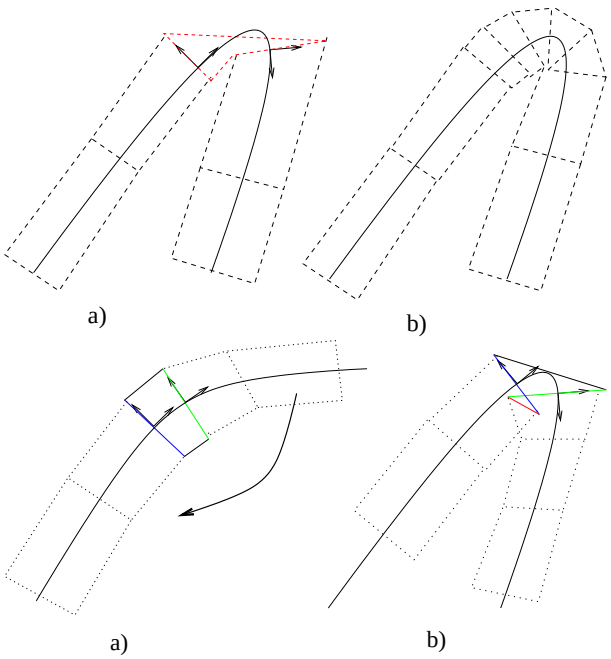


Figure 2. classical tessellation problems on generalized cylinders. Top line: inadequate sweep tessellation (left) and correct one (right). Bottom line: mesh self-intersection when deforming initial sweep.

the software-side computation, to the benefit of hardware-side, in the global rendering process. Section 5 discusses the guaranteed frame rate constraint, and presents a technique to automatically adapt the different parameters of the display process so that rendering respects a given frame rate. Finally, in section 6, some results, along with measures, are presented.

2. Generalized Cylinders

We do not plan to completely define sweeps, as they are quite a classical graphics modeling tool. We refer the reader to [8] for a classification, and extensive terms definition. In this section, we first simply set the notations we will use in this article, and give the limitation of the techniques described in this article. Second, we present classical techniques of sweep tessellation, including non-uniform axis sampling, and multiresolution techniques.

2.1. Definition and limitations

A generalized cylinder G is primarily defined by two simple analytic objects: an axis $C(u)$ (an arbitrary parameterized curve, e.g. a spline, or subdivision curve) and a cross section function $S(u)$, a planar shape the definition of which depends on the u value. For each value of u, the intersection of G with the plane orthogonal to curve C is exactly the cross section $S(u)$. We will refer to rotational primitives as the special objects where $S(u)$ is isotropic, i.e. equivalent to a circle of radius $r(u)$ (which is called profile function). Twists define G using a rotation of angle $\theta(u)$ (this function is the twisting function) around the first derivative vector $C'(u)$ direction, that is applied to an initial cross section curve S to give $S(u)$ (see e.g. [10] for simple examples). Interpolational cross-section objects are objects for which cross-sections are defined using key sections defined for key parameter values. Intermediate sections are generated as interpolation between the key cross sections [10].

Our work is primarily devoted to the (somewhat classical) case where $S(u)$ can be seen as the image of an original section S, combined with a profile curve $r(u)$, and a twisting function $\theta(u)$. Sampling technique can be generalized to interpolational objects, but hardware based rendering technique as described in this article is not correct anymore for such generalized cylinders (see section 4 for precise explanation of such a limitation). For the remainder of this article, we hence consider that our sweep is constructed using an axis $C(u)$, a cross-section S, a profile curve $r(u)$, and a twisting function $\theta(u)$.

2.2. Classical tessellation process and axis sampling techniques

Classical sweep tessellation process involves an iterative treatment. Some polygonal approximations of cross-sections $S(u)$ are created for several key u values, and two consecutive sections generate a stripset mesh information, that locally approximates the sweep. The core point of such a process is the question of the positioning of the key frames along the axis. Bloomenthal [6] presents efficient numerical techniques for calculating the reference points, and get rid of hypothetical twists of Frenet basis around the axis, that could introduce undesired and uncontrolled twists on the resulting object. The other point, about key frames positioning, is the selection of adequate parameter u values, in order to position the sampling points along the axis. Typically, it involves studying the analytic structure of axis $C(u)$, in order to isolate maximal curvature points, and give proper tessellation of the sweep. Spline curves are quite a common way to achieve axis definition [19]. For such analytic structures, many different tools exist in regard of high-curvature point isolation: interval analysis [22, 17, 4], symbolic root-isolation [11], wavelet decomposition using semi-orthogonal or biorthogonal B-spline multiresolution analysis [18, 16]. Apart from simple oversampling, inter-

val analysis provides quite an efficient way for fast partitioning of curve into parts of different interests in regard of curvature [4]. Symbolic root finding is also efficient in spline context: in most cases, spline basis functions are low degree piece-wise polynomials, or rational polynomials, and are good candidates for such numerical techniques. Wavelets provide a good theoretical framework for curve analysis, they are very flexible, and are a good way to detect high-curvature points. Yet, they are quite computation expensive, and not affordable in a high-performance rendering framework. Moreover, only B-splines and NURBS are provided with such tools, what makes it not available for all spline models. Some heuristic techniques have to be found for determining which point can be ignored when willing to simplify a curve, and the determination of such heuristic is more or less simple, depending on the B-spline structure one uses (we refer the reader to [16, 15] for a more detailed explanation of this technical point).

It is to note that curves defined using subdivision processes can somewhat take advantage of the same tools as splines, as it is possible, using theoretical impulse response of the subdivision filters, to consider subdivision curves in a formalism quite similar to that of splines.

Next section discusses the algorithm we used for efficient spline high-curvature detection, and the associated axis sampling process.

3. High curvature detection and adaptive sampling

The underlying idea behind the sampling process we discuss here is somewhat fairly simple. Most interesting spline models have the regularity property [21]. This analytic property has among its consequences that no undesired oscillation appears in a given spline segment. In other words, a curve overall shape can somewhat be pre-determined by studying the spline control points configuration. In this section, we consider a spline axis C. In our tests, we used uniform cubic B-spline axis, but most of the technique can be easily generalized to many other spline cases, as most of the spline properties used remain available for most of classical models.

3.1. Algorithm description

the sampling process uses the control points sequence $(P_0, P_1, \ldots, P_n)$ of C, the corresponding key parameter values $(u_0, u_1, \ldots, u_n)$ (e.g. knot vector for B-splines model), and some threshold value ε, as entry variables. The sampling algorithm is split into two steps:

- A sequence of scalar values $\{c_i\}_{i=0\ldots n}$ is defined using (in the following equation the notation . stands for the standard euclidian scalar vector product):

$$c_i = \overrightarrow{P_{i-1}P_i}.\overrightarrow{P_iP_{i+1}}$$

This sequence is then normalized:

$$\forall i, c_i \longleftarrow \frac{c_i}{max_{k=0\ldots n}(|c_k|)}$$

.

Extremal index i values are treated using ghost points defined using Bessel technique [12]. Each spline segment connection is hence associated to a numerical value c_i that measures the local control point "perturbation". It is to note that, depending on the spline model one uses, extremal indices for the $\{c_i\}_{i=0\ldots n}$ sequence might have to be slightly modified (e.g. extremal B-splines): such possible modifications can be easily done without loss of generality. Figure 3 shows a visual evaluation of this step, associating different color to each segment, depending on the numerical values obtained for scalar product.

Figure 3. An example of spline segment coloration using our curvature isolation process, during interactive sweep deformation. Darker segments correspond to those that the algorithm selects as high curvature ones.

- For all the parameter segments $[u_i, u_{i+1}]$, if $|c_i| < \varepsilon$ and $|c_{i+1}| < \varepsilon$, then the spline segment $[u_i, u_{i+1}]$ is treated as a whole axis sample by display (see section 4.2 for details about this treatment). In any other case, a symbolic root extraction is performed. In the case of uniform cubic B-splines, the maximal curvature parameter point on a segment satisfies to a linear equation. For all piecewise polynomial splines, such an

equation can be written; in most practical cases, the polynomial spline degree is low enough so that such equation can be solved symbolically. Once this maximal curvature m_i parameter point is calculated, then the parameter segments $[u_i, m_i]$ and $[m_i, u_{i+1}]$ are displayed, according to the technique described in section 4.2.

3.2. Comments

The regularity property ensures that high curvature segments are detected within the first pass of the process. It only involves simple and fast computation, and fits requirement for real-time computation. The second pass positions sampling points where necessary, in order to solve the problems mentioned on Figure 2. This sampling process is somewhat quite similar to that described in [11]. The main difference lies in the simplification done in order to have high-performance, relaxing theoretical validations of the sampling scheme: in particular, our scheme does not guarantee the minimality of the sample number in regard of any error criteria. It simply ensures that no high curvature point will be missed. This process makes that a spline is turned into a set of particularized points along the curve: all the parameter values corresponding to segment extremities, and additional parameter values m_i where high-curvature has been isolated. the resulting parameter segments are displayed using hardware-based technique, described in the next section.

4. Hardware based rendering using skinning

The tessellation technique we use is based on the skinning extension of most recent 3D cards [2].

4.1. Skinning principle

This OpenGL extension (sometimes called *vertex blending*) is classically devoted to character deformation, and is presented by classical documentation as a tool that allows for continuous deformation of a mesh defined as a skin over an animated skeleton. The principle of this extension is the following: it is classical to define a simple geometric transformation, using some 4x4 matrix, that defines a composition of rotation, translation, and scale, and allows for an initial mesh to be positioned anywhere within a given scene, along with simple deformations. Skinning uses, for an object, two matrices (i.e. two different positions of the object), and for all the vertices of the object, a scalar value (called *weight*) that is used to interpolate the vertex position. Precisely speaking, given an initial vertex ν^0, of weight $\omega(\nu^0)$, and two transformations M_1 and M_2, the resulting position ν of the vertex that is used for display is given by :

$$\nu = \omega(\nu^0)M_1\nu^0 + [1 - \omega(\nu^0)]M_2\nu^0$$

.

Such transformation is done by hardware, given the two matrices for a given mesh, and the weights for the mesh vertices. No CPU computation is involved in this process. Setting optimal weights for a given deformation is, in general, a difficult question. Bloomenthal presents a technique for automatically setting weights when skinning is used to character animation [7]. Such a transformation process allows for simple deformation of initial mesh. Figure 4 shows a simple 2D example of what is obtained using initial rectangle deformation, using weights that only depend on the position z along the rectangle axis. On this figure, the weight function has the same structure as blending functions described for other purposes in [5]. Controlling the weight distribution curve gives control on the way the 3D card will interpolate from the first transformation to the other.

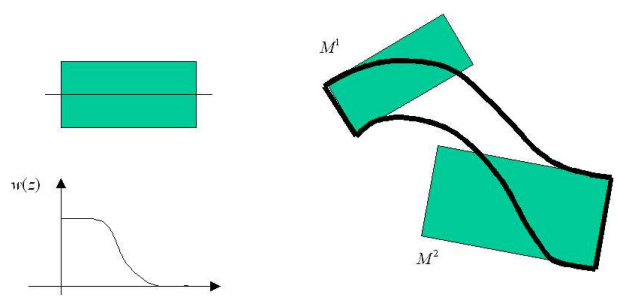


Figure 4. Simple skinning application to a simple polygonal primitive. Combination of M_1 and M_2 geometric transformations on the object on top-left, using the weights of bottom-left side, gives the result drawn in bold (right).

4.2. Adaptation to generalized cylinders visualization

Such a definition gives quite powerful tessellation elementary cell: as we are interested in generalized cylinders defined using a constant profile (see section 2.1), we can use a more complex tessellation primitive than traditional polygon. Figure 5 shows a 3D example of tessellation elementary cell: those deformed cylinders are generated using a constant, linear, cylinder, and weight function similar to that of figure 4; only M_2 transformation matrix varies throughout the deformations. It is important to note that

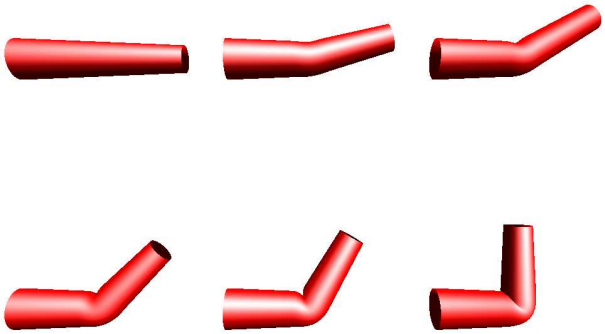


Figure 5. Example of tessellation primitive (simple rectilinear cylinder) combined with different M_2 transformations using vertex blending. Picture extracted from [2].

all those deformations are generated using exactly the same initial cylinder.

In the general case, when the profile is not a circle, we use as display primitive a linear extrusion of the initial profile. Combining this simple, precalculated, shape with the deformation process described above allows us to significantly reduce the software side of the tessellation process, and reduce it to transformation matrices computation. This is the key point in our tessellation process, described in details below. The overall tessellation algorithm can be seen as follows:

- the sampling process (see section 3) gives a set of parameter intervals.
- for each of the intervals generated by algorithm described in section 3, two frames (one for each segment extremity) are calculated along the axis C, using tools given in [6]. Each frame gives local orientation of the sweep cross-section. Combined with profile function value $r(u_i)$, and twist function $\theta(u_i)$, we have local rotation, translation, and scale necessary to position our tessellation primitive, and generate M_1 and M_2 for the considered parameter interval.

Figure 6 shows an example of tessellation achieved using this technique, using circular section (hence, the display primitive is a deformed cylinder). One can see that high-curvature points are handled in a much better way, and that the overall tessellation is somewhat better than that of figure 1. The software part of this tessellation only involves calculus of geometric transformations, and no vertex position is explicitly evaluated and transmitted to the 3D card during the display process. Since the tessellation primitive is constant throughout the time (only matrix changes from

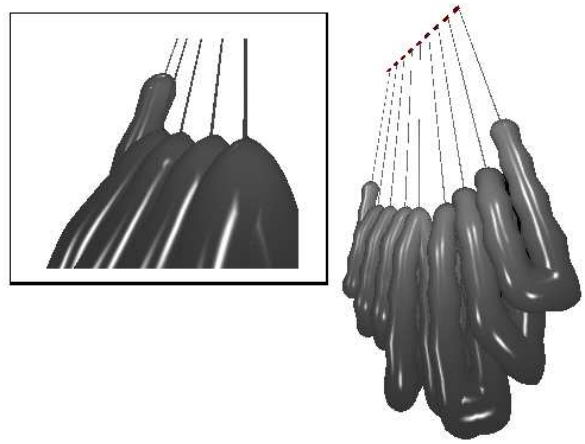


Figure 6. High quality rendering using vertex weight. Top-left: close-up on high curvature points. To be compared with tessellation created in Figure 1.

a parameter interval to another), it can be stored in a *displaylist*, using *vertexArray* technology [2], in order to take full advantage of graphical data transmission optimizations.

4.3. Comments

A few points are necessary to comment. First, the use of skinning explains the limitation mentioned in section 2.1, about the type of generalized cylinders this visualization technique can handle. It is mentioned above that the tessellation primitive is combined with axial deformations. In order to achieve this, it is necessary that the cross section did not change continuously throughout the generalized cylinder.

Second, it is also to note that there is no mathematical proof available that such a deformation tool can properly handle the classical tessellation problems shown on figure 2. Yet, as one can see on figure 6, the visual results are significantly improved by this technique.

It is also to note that raising the number of vertices on the tessellation primitive provides smoother deformation of it (which is quite natural). Yet, this smoothing involves only small computation overhead, as such deformation process is totally performed by hardware. The only overhead is the one needed by transmission of the tessellation primitive to the graphic card. This is one of the key points of the next section.

5. Guaranteed frame-rate

The whole visualization process described above involves two parameters, that makes the visualization more or less accurate. Naturally, the faster the process, the less accurate it is. Threshold value ε (see section 3) determines how many parameter intervals will be used (this number is at least the number of spline segments on the axis C). The complexity of the primitive mesh used for tessellation also determines the quality of the overall shape: the coarser the primitive is, the coarser the deformation performed by the 3D card is. This complexity can be defined using an integer n. In our implementation, we actually use several versions of the same tessellation primitive (in our example, a cylinder). The number of polygons on each version is a $O(2^i)$ for $i = 0, \ldots, n$. For a given parameter interval, raising the number of vertices on the tessellation primitive produces smoother interpolation, with only a little measured computation overhead.

The tessellation system is combined with an algorithm that compares the current framerate to a reference framerate, and adapts automatically the ε value, and complexity n of the primitive used. The global process uses several representations of the tessellation primitives, from coarser one to finest. The principle of the algorithm is fairly simple. Actually it can be seen as a geometric version of iterative energy minimization processes [1]: considering the distance between measured framerate and needed one, tuning functions are applied to ε and n. Figure 7 illustrates this.

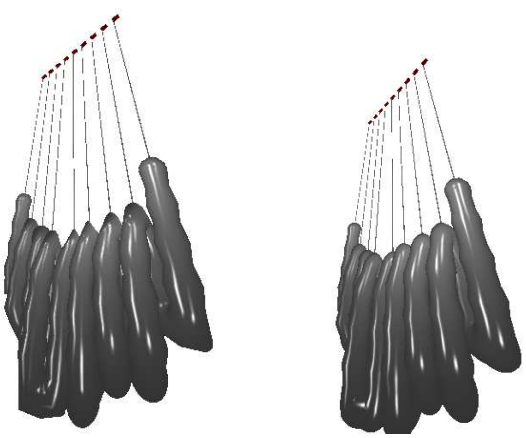


Figure 7. low quality tessellation (left) and high quality tessellation (right) obtained using our automatic frame-rate adaptation system. Respectively 900 and 460 frames/sec, on a GeForce4 based system.

Precisely speaking, the iterations from one display configuration (ε_k, n_k) to the next one $(\varepsilon_{k+1}, n_{k+1})$ uses the following relations:

$$\begin{aligned} \varepsilon_{k+1} &= \varepsilon_k + \alpha.(f_0 - f_k) \\ m_{k+1} &= m_k + \beta(\varepsilon_{k+1} - \varepsilon_k) \\ n_{k+1} &= \lfloor m_{k+1} \rfloor \end{aligned}$$

Where m_k is a floating point version of the integer variable n_k, α and β are two arbitrary constants, f_k is the current measured framerate, and f_0 the desired one.

In our implementation, we used $\alpha = 10^{-4}$ and $\beta = 5$. The tessellation process stabilizes itself at the desired framerate in less than a second.

6. Tests and results

Our test code belongs to a surgical simulator [20]. The generalized cylinder rendering technique described in this paper is used for visualizing a virtual intestine, within laparoscopic surgery simulation system [13]. For our tests, we used the simulation scene presented on figure 3, where the axis shape of the cylinder is determined by physical simulation, influenced by a virtual small sphere, controlled by the user. The weight function we used is a piecewise-polynomial function defined in [5]. Two different configurations have been tested: computer 1 is bi-athlon 2000 using GeForce4, computer 2 is bi-athlon 1600 using GeForce2mx. Hardware based technique appears to run significantly better on machine 1 (actually 5 times faster according to our measures), where high quality rendering provides 460 frames per second, and adaptive sampling combined with automatic frame-rate adaptation allows for a rendering pass that takes less than a millisecond, providing the user with a quality similar to that shown on left side of Fig. 6.

7. Conclusion

In this article we presented a global technique for high-quality, high-performance rendering of highly deformable generalized cylinders, using both efficient axis sampling incremental algorithm, and hardware based visualization technique. We also introduced a self-parameterizing algorithm, that allows a static code to guarantee a given rendering speed on any machine, degrading the rendering quality when needed.

References

[1] *Numerical Recipes*. Cambridge University Press. Available at http://www.nr.com/.

[2] Nvidia programmer online reference. http://developers.nvidia.com/.

[3] G. Agin and T. Binford. Representation and description of curved objects. *IEEE Trans. On Computers*, 25(4):439–449, 1976.

[4] J. Berchtold, I. Voiculescu, and A. Bowyer. Interval arithmetic applied to multivariate bernstein-form polynomials. Tech. Report 31/98, School of Mechanical Engineering, Univ. of Bath, October 2000.

[5] C. Blanc and C. Schlick. Extended field functions for soft objects. *Implicit Surface'95 Proc.*, pages 21–32, 1995.

[6] J. Bloomenthal. Calculation of reference frames along a space curve. In *Graphics Gems (A. Glassner ed.)*, pages 567–571, New York, 1990. Academic press.

[7] J. Bloomenthal. Medial-based vertex deformation. In *Symposium on Computer Animation Proc.*, pages 147–151, San Antonio (USA), July 2002.

[8] W. Bronsvoort, P. V. Nieuwenhuizen, and F. Post. Display of profiled sweep objects. *The Visual Computer*, 5:147–157, 1989.

[9] S. Coquillart. A control point based sweeping technique. *IEEE Computer Graphics & Applications*, 7(11):36–44, 1987.

[10] B. Crespin, C. Blanc, and C. Schlick. Implicit sweep objects. *Computer Graphics Forum (Eurographics'96 Proc.)*, 14(3):165–174, August 1996.

[11] G. Elber and E. Cohen. Second order surface analysis using hybrid symbolic and numeric operators. *ACM Transaction on Graphics*, 12(2):160–178, April 1993.

[12] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, inc. second edition, 1990.

[13] L. France, A. Angelidis, P. Meseure, M.-P. Cani, J. Lenoir, F. Faure, and C. Chaillou. Implicit representations of the human intestines for surgery simulations. *MS4CMS'02 Proc.*, November 2002.

[14] C. Grimm. Implicit generalized cylinders using profile curves. In *Implicit Surface'99 Proc.*, 1999.

[15] L. Grisoni. *Elements de multiresolution en modelisation geometrique (in french)*. PhD thesis, University of Bordeaux I, 1999.

[16] L. Grisoni, C. Schlick, and C. Blanc. Hermitian b-splines. *Computer Graphics Forum*, 18(4):237–248, December 1999.

[17] J. C. Hart, A. Durr, and D. Harsh. Critical points of polynomial metaballs. *Implicit Surface'98 Proc.*, pages 69–76, 1998.

[18] R. Kazinnik and G. Elber. Orthogonal decomposition of non-uniform bspline spaces using wavelets. *Eurographics'97 Proc.*, pages 27–38, August 1997.

[19] M. Kim, T. Chang, J. Lee, and S. Hong. Direct manipulation of generalized cylinders based on b-spline motion. *The Visual Computer*, 14(5):228–239, 1998.

[20] P. Meseure and C. Chaillou. A deformable body model for surgical simulation. *Journal of visualization and Computer Animation*, 11(4):197–208, September 2000.

[21] L. Piegl and W. Tiller. *The NURBS book*. Springer Verlag, 1995.

[22] J. Snyder. Interval analysis for computer graphics. *Computer Graphics (SIGGRAPH Proc.)*, 26(2):121–130, July 1992.

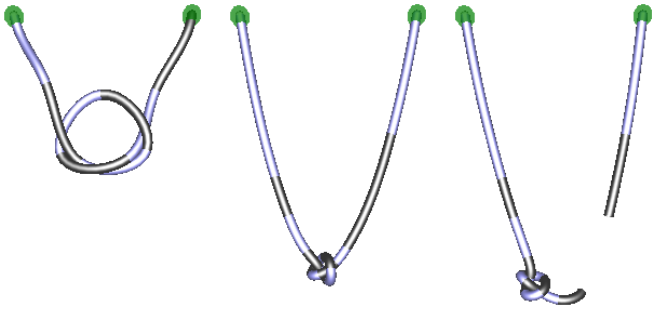
Adaptive resolution of 1D mechanical B-spline

Julien Lenoir*
CIMIT, Simulation Group

Laurent Grisoni†
ALCOVE, INRIA Futurs, LIFL

Philippe Meseure‡
SIC, University of Poitiers

Christophe Chaillou§
ALCOVE, INRIA Futurs, LIFL



(a) Fixed resolution (b) Adaptive resolution (c) Cutting

Figure 1: Knot tying and cutting

Abstract

This article presents an adaptive approach to B-spline curve physical simulation. We combine geometric refinement and coarsening techniques with an appropriate continuous mechanical model. We thus deal with the (temporal and geometric) continuity issues implied when mechanical adaptive resolution is used. To achieve real-time local adaptation of spline curves, some criteria and optimizations are shown. Among application examples, real-time knot tying is presented, and curve cutting is also pointed out as a nice side-effect of the adaptive resolution animation framework.

CR Categories: I.3.5 [Computing Methodologies]: COMPUTER GRAPHICS—Computational Geometry and Object Modeling: Splines I.3.7 [Computing Methodologies]: COMPUTER GRAPHICS—Three-Dimensional Graphics and Realism: Animation I.6.0 [Computing Methodologies]: SIMULATION AND MODELING—General I.6.8 [Computing Methodologies]: SIMULATION AND MODELING—Types of Simulation: Continuous

Keywords: Adaptive resolution, Real-time simulation, Mechanical 1D model

1 Introduction

Real thready objects have a spatial masses distribution but mostly located along a skeleton. Usually, thready model is defined as a 1D model representing the skeleton of the real object, showing most common behavior of the real object. Real time simulation implies to limit the computation time of the model. But a model also has to have a certain number of degree of freedom to be able to exhibit complex behavior. One way to deal with both limitations is to use an adaptive model that changes locally and dynamically its own resolution.

In this paper, we propose a new approach to simulate adaptive B-splines with respect to physical properties and continuity during the simulation. Our main idea is to benefit from the huge literature concerning spline's subdivisions [Finkelstein and Salesin 1994] and adapt selected methods to mechanical purposes. Our main problem is thus to properly define a mechanical model which can naturally endure both refinement and coarsening operations. We will see that this implies to define the physical properties such as mass, damping or elasticity in a continuous way. Deformations of the model are not the main issue of this paper so that we only

*e-mail: Julien.Lenoir@lifl.fr
†e-mail: Laurent.Grisoni@lifl.fr
‡e-mail: philippe.meseure@sic.sp2mi.univ-poitiers.fr
§e-mail: Christophe.Chaillou@lifl.fr

deal with the stretching deformation introducing a physical structure into the model. A Typical problem of adaptive 1D model is to tie a knot. We show as a result that our proposition permits to handle such a problem. We also point out that our technique can be use to dynamically cut a curve at any location.

This paper is organized in the following way. After the presentation of related work in section (2), we expose the spline subdivision technique and the physical simulation of splines in section (3). In section (4), we show how we combine these two techniques to get a mechanical adaptive simulation of curves. Section (5) shows results and some applications of our work before concluding the paper.

2 Related Work

Adaptive techniques are useful for 1D physical model in case of extreme deformations like knot tightening. This special manipulation of knot tying has been studied this last years. For example, [Wang et al. 2005] propose a discrete model based on punctual masses linked by a set of springs providing stretching, bending and twisting deformations. Unfortunately, the model is not adaptive, so that a random tying needs a very strong discretised thread. Another way to deal with knot tying is to use a non physics based model, like [Brown et al. 2004]. Their model is defined by a set of punctual points moving by a "follow the leader" rules. Unfortunately, this technique does not permit to take into account all physical behavior like [Wang et al. 2005] pointed out. Moreover, this model is not adaptive so that it is also subjected to a strong discretisation.

The mechanical multi-resolution technique has been studied for a few years to overcome the cost of computing the behaviour of a huge amount of points of a physical model. A coarse model is progressively refined in order to increase the number of points only where a high precision is needed, generally where interactions occur. Two main kinds of approaches have been proposed depending on whether an ideal continuous model exists or not.

The first methods do not rely on a continuous model and describe only how the resolution of a discrete model can be refined or coarsened. This process is applied depending on the needed spatial or temporal frequencies of an interaction [Luciani et al. 1995]. Such approaches have focused on spring/mass nets [Hutchinson et al. 1996; Ganovelli et al. 1999]. They consist in adding or removing points of the net, and connecting them with the points of the lower resolution. However, springs have drawbacks. Indeed, splitting a spring into several implies higher elasticity constants (and a less stable integration of the dynamics equations). In practice, a spring is usually split at its middle, and the elasticity constants are doubled. This technique keeps the resolution away from an exact adaptation, leading to an iterative process. Moreover, the distribution of masses between the implied points is not straightforward. To avoid such distribution, Eberhardt et al. propose to use virtual particles [Etzmuss et al. 2000]. [Phillips et al. 2002] used a similar model to simulate a knot tying. When the resolution change, new nodes are inserted or removed and the total mass and moment are kept. But, as springs are placed on the nodes, the configuration changes during a changing resolution so as to the behaviour.

The second family of approaches rely on a continuous "ideal" model which is discretised at different resolutions. Finite-element or finite-difference methods are used [Astley and Hayward 1997; Wu et al. 2001; Debunne et al. 2001]. Compared to the previous family, theses methods guarantee that the physical laws do not depend on the discretisation level, since they refer to the continuous model. Nevertheless, the finer the discretisation is, the more precise the behaviour becomes (levels appear as low-band filters). To allow the use of different levels depending on where the interaction occurs, the different resolution levels must be linked. Thus, these are defined by splitting elements or using higher order basis functions [Grinspun et al. 2002], which allow to control geometric continuity between elements of different levels. Debunne et al. proposed a method to link independently-built discretisation levels together [Debunne et al. 2001]. Another way to build the different discretisation levels is to use multi-resolution Free-Form Deformation [Nocent et al. 2001; Capell et al. 2002]. One of the major concerns of all these methods is to ensure continuity when a switch occurs between levels. What is more, all of these techniques usually involve some pre-computation on regular, pre-set, subdivisions of the initial model. Providing point insertion at any location on the object usually involves dense refinement around point location, and only provides approximate adaptivity. In other words, the adaptation is surely local, but not enough precisely localised.

Our approach falls into the second family. Yet, in our model, we do not have to deal with the continuity problems between elements of different levels or during a level switching. The main idea is to rely on a continuous object which is discretised by a variable number of degrees of freedom. All the physical degrees of freedom are only based on the geometry. By using methods which guarantee the geometric invariance when adding or removing degrees of freedom, we naturally ensure the temporal continuity of the shape and its behaviour. What it more, no pre-computation needs to be achieved, and as is shown in this article, B-spline multi-resolution straightforwardly provides insertion at any location. It can be point out that our technique is not a multi-resolution one as it does not deal with multiple resolutions at the same time. Our model is based on one adaptive resolution.

3 B-spline curve physical simulation

In this section we first define the notation used in the remainder of this article, and recall some of the B-spline's proper-

ties used to adapt the resolution. We then describe classical process for B-splines physical animation, and the general framework we use.

3.1 B-spline curve definition and properties

A B-spline curve is defined using classical spline definition:

$$C(s) = \sum_{i=0}^{N} q_i b_{i,d}(s) \tag{1}$$

In the above equation, q_i are the *control points* of the curve C. The functions $b_{i,d}$ are piecewise polynomials of degree d, defined using a *knot vector*, sorted list of scalar values:

$$T = (s_0, s_1, \ldots, s_M) \quad \text{with} \quad s_0 \leq s_1 \leq \ldots \leq s_M$$

From such a knot vector, classical Cox-DeBoor recursive definition[Cox 1972; de Boor 1972] is used either to evaluate, or symbolically calculate, B-spline basis functions.

B-splines have many numerical and geometrical properties [Farin 1990], among which exact knot insertion [Prautzsch 1984; Schumaker 1981]. Precisely speaking, we consider a first B-spline representation space, constructed using the known vector $U = (u_0, \cdots, u_n)$: the generated basis functions of degree d, are written here $F^0_{j,d}$. We also consider a second B-spline space constructed using some knot vector $W = (w_0, \cdots, w_{n+k})$ created from U inserting k knots, that is, we suppose that there exists some injection σ, that maps $\{0, \cdots, n\}$ into $\{0, \cdots, n+k\}$, such that:

$$\forall i \in \{0, \cdots, n\}, \begin{cases} i \leq \sigma(i) \\ u_i = w_{\sigma(i)} \end{cases} \tag{2}$$

Basis functions generated from W of degree d are written $F^1_{i,d}$. Theoretical knot insertion properties corresponds to the existence of coefficients $\beta^d_{i,j}$ such that:

$$\forall j \in \{0, \cdots, n-d\}, \forall s, F^0_{j,d}(s) = \sum_{i=0}^{n+k-d} \beta^d_{i,j} F^1_{i,d}(s) \tag{3}$$

The Oslo algorithm defines $\beta^d_{i,j}$ coefficients, using recursive formula [Prautzsch 1984]:

$$\beta^0_{i,j} = \begin{cases} 1 & \text{if} \quad u_i \leq w_j < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$
$$\text{and, for } r = 0, \cdots, k-1: \tag{4}$$
$$\beta^{r+1}_{i,j} = \frac{w_{j+r} - u_i}{w_{i+r} - u_i} \beta^r_{i,j} + \frac{u_{i+r+1} - w_{j+r}}{u_{i+r+1} - u_{i+1}} \beta^r_{i+1,j}$$

Such an insertion process will be later in this article used in its global matrix form:

$$F^0 = \beta^T F^1 \tag{5}$$

where F^0 is the vector composed of the functions $F^0_{j,d}$, F^1 the vector composed of the functions $F^1_{i,d}$, and β^T the transpose of matrix β, matrix composed of the coefficients $\beta^d_{i,j}$, evaluated using eq. (3). It is to note that, in order to make notation more readable, indexes representing spline degree d will be removed from notation in the remainder of the article (i.e $b_{i,d}$ becomes b_i, etc...), as the spline degree is always considered as constant. Among other classical consequences of knot insertion property, a curve, represented by a control point sequence q^0 on the knot vector U, will be represented on the knot vector W using control point sequence $q^1 = \beta q^0$. It is important to understand that knot insertion automatically entails control point insertion, and that the first can not be achieved without the second.

It is to note that in the cubic case, that is, without any loss of generality about the presented results, the B-spline example case we used, the insertion of a single knot value only involves two non-zero values, i.e. in case of knot at index i, $\beta_{i,i}$ and $\beta_{i,i-1} = 1 - \beta_{i,i}$. In that specific case, the new control points are given by the simple relation:

$$q^1_i = \beta_{i,i} q^0_i + (1 - \beta_{i,i}) q^0_{i-1}$$

With such an insertion property, B-splines functions are said to be *refinable*, which is the basic property for wavelet based multi-resolution availability [Finkelstein and Salesin 1994; Kazinnik and Elber 1997]. As B-spline functions are local, insertion process is guaranteed to be in linear complexity in term of spline degree, and independent on the number of control points. The question to know which points can be removed, or, on the contrary, to calculate parametric values where knot should be inserted, typically involves studying the analytic structure of $C(t)$, in order to isolate maximal and minimal curvature points. Many different tools exist for B-splines in regard of high-curvature point isolation: interval analysis [Snyder 1992; Hart et al. 1998; Berchtold et al. 2000], symbolic root-isolation [Elber and Cohen 1993], wavelet decomposition using semi-orthogonal or biorthogonal B-spline multi-resolution analysis [Kazinnik and Elber 1997]. Interval analysis provides quite an efficient way for fast partitioning of curve into parts of different interests in regard of curvature [Berchtold et al. 2000]. Symbolic root finding is also efficient in spline context: B-spline basis functions are piecewise polynomials, most of the time of accessible degree. Wavelets provide a good theoretical framework for curve analysis. Yet, they are quite computation expensive, and hardly affordable in a high-performance framework.

From a theoretical point of view, several results and techniques exist for B-spline knot removal [Eck and Hadenfeld 1995; Tiller 1992]. B-spline curve allow for exact knot removal when such operation can be inverted, i.e. shape is not modified and re-insertion of the removed knot provides exactly the original curve configuration. Since the Oslo algorithm changes existing control points location, it is natural to also expect modification of neighbour control points during knot removal. [Eck and Hadenfeld 1995; Tiller 1992] provide general algorithm for B-spline knot removal. We detail here the case we used for our tests, i.e. simple cubic case, that is simple application of these algorithms. The removal

process must be the inverse function of the insertion (see eq. (3)). We consider for the explanation a spline defined by a set of control points $q^\star$ and a knot vector $t^\star$. We also consider a refined version of this curve, defined using control point q and knot vector t, that only has one inserted point at abscissa $t_{j+d} + \delta.(t_{j+d+1} - t_{j+d})$, with $0 \leq \delta \leq 1$.

The insertion algorithm gives the following equations:

$$\begin{cases} \forall k < j, \;\; q_k = q_k^\star \\ q_j = \beta_{j,j} q_j^\star + (1-\beta_{j,j}) q_{j-1}^\star \\ q_{j+1} = \beta_{j+1,j+1} q_{j+1}^\star + (1-\beta_{j+1,j+1}) q_j^\star \\ q_{j+2} = \beta_{j+2,j+2} q_{j+2}^\star + (1-\beta_{j+2,j+2}) q_{j+1}^\star \\ \forall k > j+2, \;\; q_k = q_{k-1}^\star \end{cases} \tag{6}$$

During knot removal, q_{j+1} is suppressed and points q_j and q_{j+2} points are respectively changed into points $q_j^\star$ and $q_{j+1}^\star$. Equation (6) easily provides:

$$\begin{cases} q_j^\star = \dfrac{q_j - (1-\beta_j) q_{j-1}^\star}{\beta_j} \\ q_{j+1}^\star = \dfrac{q_{j+2} - \beta_{j+2} q_{j+2}^\star}{1-\beta_{j+2}} \end{cases} \tag{7}$$

3.2 Global animation process

The following algorithm presents the general algorithm used to simulate the material B-spline curve.

```
MECHANICAL LOOP()
while 1
  do
     COLLISIONSDETECTION()
     COLLISIONSRESPONSE()
     for (All objects o)
        do O->COMPUTEACCELERATION()
     NUMERICALINTEGRATION()
     UPDATEDATA()
```

Functions about collisions handling (i.e. collisionsDetection() and collisionsResponse()) contain very classical algorithms and are not within the scope of this section, as they will no further be discussed in the remainder of the article. We refer the reader to [van den Bergen 2003] for possible techniques. For numericalIntegration() function, many techniques are also available in literature [Witkin and Baraff 1997]. The main idea of this function is to determine the new positions and velocities of the spline's control points from their acceleration. Each model just has to compute its accelerations and update its data at the end of the mechanical iteration. Only content of computeAcceleration() function is discussed in this section.

To be able to animate the curve realistically, a physical simulation is applied to it, which aim at computing the successive positions of all the control points. To allow multi-resolution, it is desirable to exploit the continuous property inherent to the curve. So we avoid using a mass/spring model, but choose a Lagrangian approach instead. This method only requires the model to define a finite set of degrees of freedom and express its energies as a function of them. Our curve is based on equation (1), thus the degrees of freedom are the control point coordinates. The lagrangian equation of the system can be derived into a linear system [Lenoir et al. November 2002]:

$$\begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{pmatrix} \begin{pmatrix} \mathbf{A^x} \\ \mathbf{A^y} \\ \mathbf{A^z} \end{pmatrix} = \begin{pmatrix} \mathbf{B^x} \\ \mathbf{B^y} \\ \mathbf{B^z} \end{pmatrix} \tag{8}$$

with

$$M_g = \begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{pmatrix} \tag{9}$$

where $\mathbf{A}$ the degrees of freedom's accelerations, $\mathbf{B}$ a vector that sums the different contributions of all forces and internal energies, and finally M_g the generalised mass matrix (resulting from the kinetic energy term) defined via the sub-matrix M of size $n \times n$:

$$\forall (i,j) \in \{1..n\} \times \{1..n\}, \quad M_{ij} = m. \int_{\mathbb{R}} b_i(s).b_j(s)\, ds \tag{10}$$

with m the mass of the spline.

The resulting system size is $3n \times 3n$ where n is the number of control points. For reasonable value of n, the simulation can be done in real time thanks to a LU decomposition of the M matrix [Lenoir et al. November 2002].

All interaction forces are inserted into the system by the way of $\mathbf{B}$. Moreover, the gravity is also taken into account by this vector via the formula: $B_i^\alpha = B_i^\alpha + m.\mathbf{g}^\alpha \int_0^1 b_i(s) ds$ with $\alpha \in \{x,y,z\}$. Also an internal energy of deformation E, which quantifies the amount of energy needed to deform the body, must be added in the vector $\mathbf{B}$. This is done by computing the variation of this energy relatively to each degree of freedom: $B_i^\alpha = B_i^\alpha - \frac{\partial E}{\partial q_i^\alpha}$. In other words, this energy tends to make the model behave in a homogenous way.

This energy can be, for example, defined thanks to springs placed along the curve and giving it some elasticity in stretching or bending. Yet, such a discrete approach is not well suited to a changing resolution stage. A more convenient way is to define a continuous energy, which is based on the shape of the curve and not a finite set of points. This type of energy takes advantage of the natural model's continuity to be independent of the B-spline resolution. On a 1D model, three types of deformation are interesting: stretching, bending and twisting [Terzopoulos et al. july 1987]. As an example, the stretching deformation part is considered in this paper as it provides consistency to the model. The main idea is just to show how the adaptive process interacts with a deformation process.

3.3 Continuous stretching energy definition

Deformation energy is classically expressed by the way of a tensor product between a strain tensor and a stress tensor.

The strain tensor measures the deformation of the body while the stress tensor computes the generated forces.

In the case of linear elasticity, Nocent and Remion [Nocent and Remion 2001] propose the application of the Green-Lagrange strain tensor to the spline curve simulation for high deformation. By developing the equations, we obtain the resulting energy term:

$$E(t) = \frac{Y.r}{8} \int_{begin}^{end} (\gamma(s,t)^2 - 1)^2 . \frac{\partial l_0(s)}{\partial s} ds \qquad (11)$$

with t the time, Y the Young modulus of the material, r the area of a curve's section (considered as constant), *begin* and *end* the valid boundaries of the parametric abscissae (for more readability, we suppose that this boundaries are respectively 0 and 1), l_0 the length of the curve in rest state and finally $\gamma(s,t) = \frac{\partial l(s,t)}{\partial s} / \frac{\partial l_0(s)}{\partial s}$ with $l(s,t)$ the current length of the curve (so that $\gamma(s,t)$ express the local dilatation factor in stretching).

To introduce this energy in the system, its first derivative with respect to the each degree of freedom q_i^α has to be computed [Nocent and Remion 2001]:

$$W_i^\alpha(t) = \frac{Y.r}{2} \sum_{m=1}^{n} q_m^\alpha(t)[B_{im} - \sum_{p,q=1}^{n} B_{impq}(q_p.q_q)] \qquad (12)$$

where

$$B_{im} = \int_0^1 b_i'(s) b_m'(s) / \frac{\partial l_0(s)}{\partial s} ds$$

$$B_{impq} = \int_0^1 b_i'(s) b_m'(s) b_p'(s) b_q'(s) / \frac{\partial l_0(s)}{\partial s}^3 ds$$

We state that such an energy can be used in a real-time context. Indeed, even if these terms can not be formally computed, they can however be accurately evaluated since they are constant over time for a given spline configuration. Moreover, the locality of the spline model permits to eliminate some computation in equation (12). All sum terms in the equation are controlled and constant (linked to the locality parameter) boundaries. In consequence, the complexity of this continuous energy computation is $\mathcal{O}(n)$.

4 Adaptive model

Our main idea is to apply the subdivision principles detailed in section (3.1) to the spline animation of section (3.2). In this section, we are interested in how the model is altered when adding or removing a control point and when such operations must be applied. Global adaptive process is first described, and then energy factors update, as well as knot insertion/removal criteria, are discussed.

4.1 Global animation algorithm description

The adaptive aspects are treated in the mechanical process just after the numerical integration stage in order not to disturb the numerical integration which calls the acceleration computation of all objects:

```
ADAPTIVE MECHANICAL LOOP()
while 1
  do
     ...
     NUMERICALINTEGRATION()
     ADAPTATION()
     UPDATEDATA()
```

the function adaptation() is organised as followed:

```
ADAPTATION()
while mustWeInsertPoint()
  do insert knot and control point
     modify control points positions and velocities
     update matrices and vectors
     update energy structure

while mustWeRemovePoint()
  do remove knot
     modify control points positions and velocities
     remove control point
     update matrices and vectors
     update energy structure
```

After each numerical integration step, adaptive criteria (see section (4.3)) are tested on the curve: curve resolution is changed accordingly, and corresponding terms (especially regarding mass matrix M, and energy expression) are recalculated.

The generalised mass matrix M (cf. equation (10)) is modified as well as its LU decomposition. As B-spline have locality property, most of the coefficient stay unchanged, and matrix M is still a band structure of width $2d+1$. The multi-resolution process, for one knot insertion/removal, affects $d+2$ basis functions and for each of them, $2d+1$ basis functions have a non null intersection support with it. Using the fact that M is symmetrical, knot insertion/removal involves $(d+2)(2d+1)$ re-computation of terms within matrix M, and the cost of such an operation is thus independent from the spline curve complexity. Yet, LU decomposition of this matrix needs to be recomputed. We presently do not update the LU decomposition but compute it completely. In practice, it is not a real drawback, since the number of points is rather small and adaptation provides a good means to keep this number small. However, if an application should need a high number of degrees of freedom, an update of the LU decomposition should be proposed. Besides, it is important to understand that the computation of a new mass matrix does not imply a physical change of the model, since it is always related to the same continuous mass distribution along the curve.

The gravitation is also re-computed using relation $B_i^\alpha = m.g^\alpha . \int_0^1 b_i(s) ds$. Again, thanks to the B-spline locality, only the $d+2$ basis functions affected by knot insertion/removal are recomputed by this formula. This update is done once

again in $\mathcal{O}(1)$ in regard of number of control points. Deformation energy must be also re-computed. As this term deals with the behavior of the model, we focus on this computation in the next section.

4.2 Adaptation of the stretching energy

The insertion at a specific abscissa affects the basis functions which supports contain this abscissa. For spline degree equal to d, this affects $d+2$ basis functions.

Changing resolution affects the stretching energy (cf equation (12)) via the terms B_{im} and B_{impq}. In these two terms, the factor $\frac{\partial l_0(s)}{\partial s}$ only depends on the original spline resolution. As a result, only the basis functions first derivative affect the two terms during a resolution change (either knot insertion, or removal) stage. The two addressed terms are updated slightly differently:

- As the expression of B_{im} is symmetrical in regard of indexes i and m, we only store B_{im} for i and m such that $i \geq m$. For a given value of i, there are $2d+1$ consecutive values of m for which B_{im} is non-null, because of the spline locality property. Using the commutativity property, useful value for m only involves $d+1$ values. And, as there are $d+2$ basis functions affected by the insertion, the total number elements to be recomputed is $(d+2)(d+1)$. As a result, the update cost of terms B_{im} is independent from spline complexity for a given knot insertion/removal.

- Again, as the expression of B_{impq} is symmetrical in regard of indexes i, m, p and q, we only store B_{impq} for i, m, p and q such that $i \geq m \geq p \geq q$. The recomputed elements on knot insertion/removal are these which have at least one index (between i, m, p and q) matching an affected basis functions's index. Once again, it is easy to see that B-spline locality entails modification cost that is only dependant on the spline degree that is used. In order to minimize coefficients storage cost, we describe here indexing technique we used, that stores exactly the needed coefficients. Indeed, B_{impq} coefficients are included in a collection that collects many null values, and heavily symmetrical (all permutations of indexes i,m,p,q provide the same value): storing such coefficients with a n^4 sized array would be wasteful. The choice we made is to use the following value indexing process: we first use, for indexing B_{impq} (with $i \geq m \geq p \geq q$), the integers q, $p-q$, $m-p$, $i-m$. This, because of spline locality, provides, for non-null B_{impq} values, a set of four integers that belong to $\{0,\cdots,n\} \times \{0,\cdots,d\}^3$, and that can quite easily be enumerated: as a result, coefficients are stored in an array, which size is linear in regard of the number of control points.

Since our continuous stretching energy only depends on the shape of the curve and since the insertion or removal of a point does not imply any shape alteration (see section (3.1)), we are assured that deformations are continuous relatively to the time.

4.3 Adaptation criteria

There are two different cases when point should be inserted on the spline curve during interactive manipulation:

- The first case is purely geometric. High curvature points on B-splines curve naturally involve mechanical limitation due to spline resolution. As a result, an insertion is needed. [Grisoni and Marchal 2003] presents a curvature evaluator adapted to high-performance, that we need here. The i-th spline segment is assigned the following value:

$$c_i = \frac{1}{2}\left(\frac{\overrightarrow{q_{i-1}q_i}.\overrightarrow{q_iq_{i+1}}}{\|\overrightarrow{q_{i-1}q_i}\|\|\overrightarrow{q_iq_{i+1}}\|}+1\right) \tag{13}$$

  Such an expression practically appears to provide fast criterion for B-spline segment maximal curvature evaluation: each spline segment connection can be, when needed, associated with c_i that measures the local control point *disturbance*. The curvature criterion of equation (13) is used for isolating high-curvature segments: a knot value (hence, a control point on the spline curve) is inserted at the middle of the i-th segment if and only if c_i and c_{i+1} are greater than some pre-set threshold value.

- The second case is more mechanically involved. In practice, for a spline curve with a constant, coarse, resolution, it is not possible to plausibly replicate action one would have on real physical model, simply because of resolution limitations. For example, acting on an elastic model at any point would intuitively entails local deformation. Such a deformation on a coarse resolution curve is not possible to simulate, because local degrees of freedom density is not high enough. As a result, we chose to insert some point each time user interacts directly on the spline curve, the new knot value being inserted at the parameter value where the action is located.

The removal of a control point is more simple. In practice, the curvature criterion defined in equation (13) is once again used, but this time in order to detect low-curvature segment. A point is removed when local curvature is low enough. Precisely speaking, control point q_i is removed when c_i is below some pre-set threshold.

It can be pointed out that the insertion process can occur anywhere without any condition. But the removal process as to verify a geometric condition to be an exact algorithm[Eck and Hadenfeld 1995]. We approximate this condition by the

low curvature criterion. By this way, we are able to tune the exactitude of the removal process. For an exact process, points are removed only when the curve is straight, which happens essentially at an equilibrium state of the simulation.

5 Applications

All the presented tests have been performed on a PIV 2.4Ghz 512MB using fast implicit Euler integration scheme and a virtual time step fixed to 1ms. Times announced in the different sub-sections are the mechanical computation times and thus only take into account the mechanical computation (including the collision process and the numerical integration stage), independently from visualisation time.

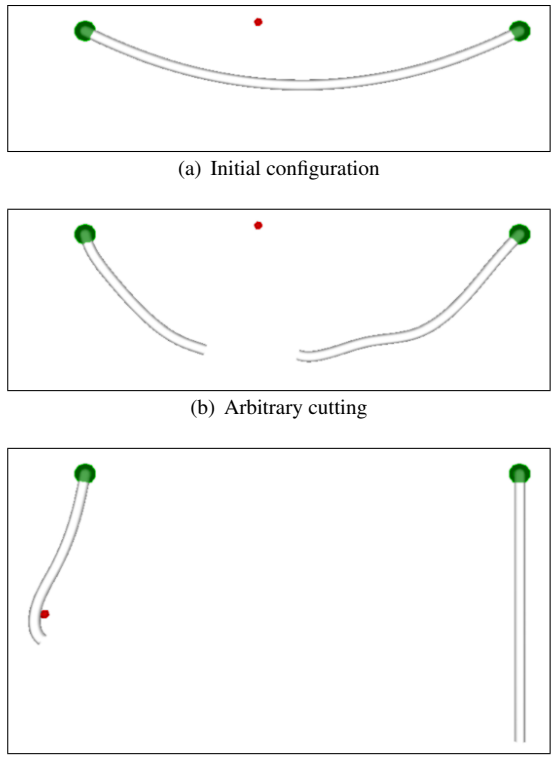
5.1 Application: B-spline curve cutting

It is known that B-spline continuity at a given knot value equals the difference between polynomial degree and knot multiplicity. This explains, among other, that cubic B-spline are the B-spline of lower degree that allow C^2 continuity. It also has as a consequence that creating knot multiplicity equal to $d+1$ for a spline of polynomial degree d creates C^{-1} discontinuity, which means that the spline curve, even when defined as a whole, will practically exist as two separate pieces. As shown here, this property directly extends to adaptive physical B-spline simulation. Knot insertion is used to simulate curve cutting.

This operation is presented by figure (2) where an initial spline of 12 control points is left hanging from its extremities (figure (2(a))). Figure ((2(b))) shows the consequence, on the simulation, of multiple knots insertion at arbitrary parametric abscissa. Figure (2(c)) shows mechanical independence of the two created pieces, a manipulation on the first piece, does not affect the simulation of the second one. This simulation takes about 4ms for each simulation step at the beginning of the simulation (12 control points and 6 constraints to fix the two extremities). At the end, the simulation step takes about 8ms with 16 control points and still 6 constraints.

Figure (3) points out the possibility to create multiple independent pieces. A curve is simulated with 10 control points and 3 different cuttings are applied during the simulation. The computation time begins at 1ms and finish around 20-30ms depending on collisions (for 22 control points). Each piece interacts independently as shown in figure (3(d)).

We stress that this cutting procedure is only a direct application of the adaptive animation technique, and involves no other special treatment (such as re-meshing) in the contrary of other cutting techniques. This simple example illustrates that the use of adaptive physically-based splines can be a powerful tool.



(a) Initial configuration

(b) Arbitrary cutting

(c) Interaction on one piece

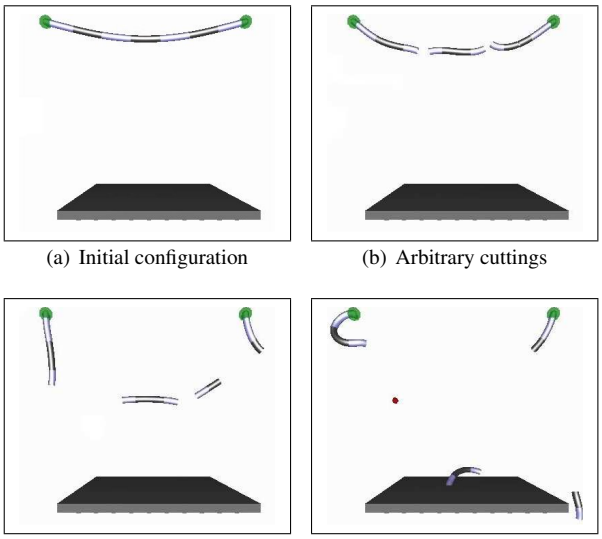
Figure 2: Example of curve cutting using mechanical B-spline multi-resolution.

5.2 Application: knot tying

The second result demonstrates efficiency of adaptive B-spline physical simulation on the classical case of knot tying. The initial spline configuration is composed with a preformed knot and the two extremities are fixed. The spline is then animated using simple gravity, in order to let the knot tie under simple action of curve weight. Figure (4(a)) shows the result of such animation with a fixed resolution and figure (4(b)) shows the result with an adaptive resolution. Figure (4(c)) zoom in the knot to be able to see the different insertions in the knot area. On each left figure, spline segments color have been alternatively changed, so that knot repartition (and implicitly knot insertion/removal) could be visible on such pictures. On each right figure, control points are shown.

On figure (4(a)), one can easily see uniform sampling limits. The curve encounters a barrier due to the lack of control implied by the limited number of degrees of freedom. This spline requires 2.27ms computing time per mechanical iteration. For the knot to be tightened, one would need a much higher number of regularly distributed control points, in order to get a sufficient control wherever the knot is tightened on the curve. In practice, this leads to prohibitive computation time for real-time purposes (several seconds or even minutes per simulation step).

On figures (4(b)) and (4(c)), the same simulation is done



(a) Initial configuration (b) Arbitrary cuttings

(c) 4 independent pieces (d) Evolution of each pieces

Figure 3: Example of curve cutting in multiple piece.

using an adaptive resolution. One can see that both knot removal (see low curvature areas of the curve) and knot insertion have been used for providing the shown result. The non adaptive spline involves 11 control points, and the adaptive one involves, at the end of the animation, 16 points. A mechanical iteration takes about 9.89ms.

The modifications, involving by the adaptive process, affect the B-spline parametrization by the way of its knot vector. At the beginning of the simulation, this vector is:

$$T = \{ \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 & 14 & \} \end{array}$$

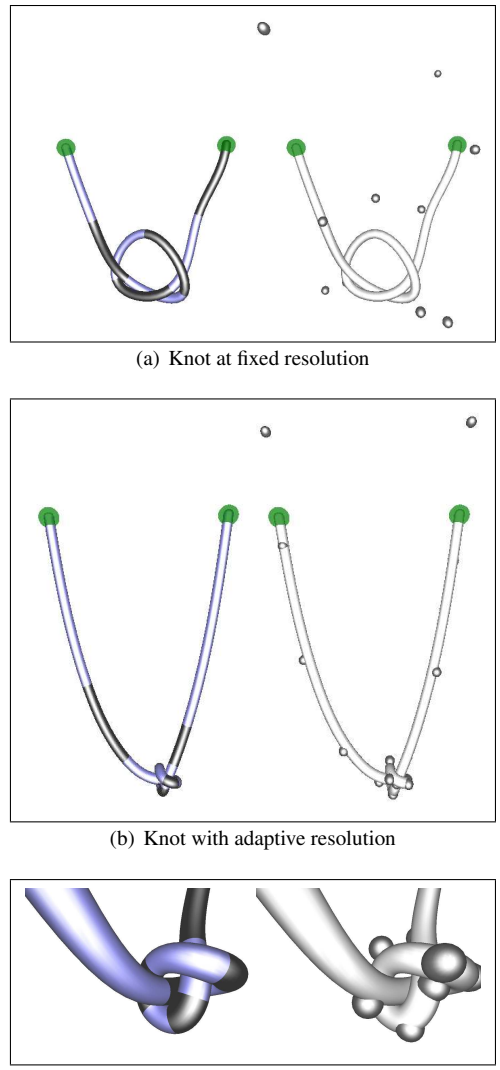
During the simulation, 9 control points are inserted (at the parametric abscissae 8.5, 5.5, 6.5, 7.5, 6.75, 7.75, 6.875, 7.25 and 7.625) followed by 4 control points removed (at the parametric abscissae 4, 5.5, 9 and 10). The removal process usually occurs when the simulation is stable and some spline segments are quite aligned. These adaptive modifications result in the following knot vector at the equilibrium state:

$$T = \{ \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 5 & 6 & 6.5 & 6.75 \\ 6.875 & 7 & 7.25 & 7.5 & 7.625 & 7.75 & 8 & 8.5 \\ 11 & 12 & 13 & 14 & \} \end{array}$$

The spline knot vector shows the locality of the process. It presents large parametric segments on low curvature locations (before and after the knot) and smaller one on high curvature case (in the knot area).

6 Conclusion and future work

We have presented in this paper an adaptive model of B-spline physical simulation with continuous deformation energy allowing a continuity in physical movement during the resolution changing. This method refines the model locally around any arbitrary point on the curve and allows a finer geometric adaptation, by the way of a higher number of degrees of freedom. This model presents interesting results for



(a) Knot at fixed resolution

(b) Knot with adaptive resolution

(c) Zoom in the knot with adaptive resolution

Figure 4: Example of knot tying.

some applications like simulation of curve cutting (for example during a surgical simulation) or knot tying.

This work could be improved by studying new terms like bending and twisting (continuous) energies permitting to treat specific curves as surgical threads. Another related work is the treatment of self collision by robust methods as Lagrange multipliers instead of penalty methods thanks to dedicated constraints.

It is also interesting to point the fact that, in order to provide efficient adaptive physical simulation, both geometric and mechanical aspects need to be taken into account. However, the link between geometric adaptation and the need for a mechanical refinement is obviously less simple than one could expect at first, and would also deserve closer study.

References

ASTLEY, O., AND HAYWARD, V. 1997. Real-time finite elements simulation of general visco-elastic materials for haptic presentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

BERCHTOLD, J., VOICULESCU, I., AND BOWYER, A. 2000. Interval arithmetic applied to multivariate bernstein-form polynomials. Tech. Report 31/98, School of Mechanical Engineering, Univ. of Bath, October.

BROWN, J., LATOMBE, J., AND MONTGOMERY, K. 2004. Real-time knot-tying simulation. *The Visual Computer 20*, 2-3 (May), 165–179.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIC, Z. 2002. A multiresolution framework for dynamic deformations. In *ACM Symposium on Computer Animation*.

COX, M. G. 1972. The Numerical Evaluation pf B-Splines. In *J. Inst. Mathematics and Applications*, vol. 10, 134–149.

DE BOOR, C. 1972. On Calculating with B-Splines. *J. Approximation Theory 6*, 1 (July), 50–62.

DEBUNNE, G., DESBRUN, M., CANI, M., AND BARR, A. 2001. Dynamic real-time deformations using space and time adaptive sampling. In *SIGGRAPH'01 Conference Proceedings, Computer Graphics annual conference series*.

ECK, M., AND HADENFELD, J. 1995. Knot removal for B-spline curves. *Computer Aided Geometric Design 12*, 3, 259–282.

ELBER, G., AND COHEN, E. 1993. Second order surface analysis using hybrid symbolic and numeric operators. *ACM Transaction on Graphics 12*, 2 (April), 160–178.

ETZMUSS, O., EBERHARDT, B., AND HAUTH, M. 2000. Collision adaptive particle systems. In *Pacific Graphics'2000 Conference*.

FARIN, G. 1990. *Curves and Surfaces for Computer Aided Geometric Design*, inc. second ed. Academic Press.

FINKELSTEIN, A., AND SALESIN, D. H. 1994. Multiresolution curves. *Computer Graphics 28*, Annual Conference Series, 261–268.

GANOVELLI, F., CIGNONI, P., AND SCOPIGNO, R. 1999. Introducing multiresolution representation in deformable object modeling. In *Spring Conference on Computer Graphics*, 149–158.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. Charms: a simple framework for adaptive simulation. In *SIGGRAPH'02 Conference Proceedings, Computer Graphics annual conference series*, 281–290.

GRISONI, L., AND MARCHAL, D. 2003. High Performance Generalized Cylinder Visualization. In *Proc. Shape Modeling International '03*, IEEE Computer Society Press, 257–263.

HART, J. C., DURR, A., AND HARSH, D. 1998. Critical points of polynomial metaballs. *Implicit Surface'98 Proc.*, 69–76.

HUTCHINSON, D., PRESTON, M., AND HEWITT, T. 1996. Adaptive refinement for mass/spring simulations. In *EUROGRAPHICS Workshop on Animation and Simulation*, 31–45.

KAZINNIK, R., AND ELBER, G. 1997. Orthogonal decomposition of non-uniform bspline spaces using wavelets. *Eurographics'97 Proc.* (August), 27–38.

LENOIR, J., MESEURE, P., GRISONI, L., AND CHAILLOU, C. November 2002. Surgical thread simulation. *Modelling and Simulation for Computer-aided Medecine and Surgery*.

LUCIANI, A., HABIBI, A., AND MANZOTTI, E. 1995. A multiscale model of granular materials. In *Graphics Interface'95 Conference*, 136–146.

NOCENT, O., AND REMION, Y. 2001. Continuous deformation energy for dynamic material splines subject to finite displacements. *Eurographics CAS'2001*.

NOCENT, O., NOURRIT, J., AND RÉMION, Y. 2001. Toward mechanical level of detail for knitwear simulation. In *WSCG'01 Conference*, 252–259.

PHILLIPS, J., LADD, A., AND KAVRAKI, L. 2002. Simulated knot tying. In *International Conference on Robotics and Automation*, 841–846.

PRAUTZSCH, H. 1984. A short proof of the oslo algorithm. *Computer Aided Geometric Design 1*, 95–96.

SCHUMAKER, L. L. 1981. *Spline Functions: Basic Theory*. John Willey & Sons, New York.

SNYDER, J. 1992. Interval analysis for computer graphics. *Computer Graphics (SIGGRAPH Proc.) 26*, 2 (July), 121–130.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. july 1987. Elastically deformable models. *Computer Graphics*.

TILLER, W. 1992. Knot-removal algorithms for nurbs curves and surfaces. *CAD 24*, 8, 445–453.

VAN DEN BERGEN, G. 2003. *Collision Detection*. Morgan Kaufmann, November. ISBN:155860801X.

WANG, F., BURDET, E., DHANIK, A., POSTON, T., AND TEO, C. 2005. Dynamic thread for real-time knot-tying. In *World Haptic Conference*, 507–508.

WITKIN, A., AND BARAFF, D. 1997. Physically based modeling: Principles and practice. In *Siggraph '97 Course notes*.

WU, X., DOWNES, M., GOKETIN, T., AND TENDICK, F. 2001. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Computer Graphics Forum 20*, 3 (Sept.), 349–358.

Time-Critical Animation of Deformable solids

J. Dequidt
LIFL
dequidt@lifl.fr

D. Marchal
LIFL - INRIA
marchal@lifl.fr

L. Grisoni
LIFL - INRIA
grisoni@lifl.fr

Abstract

This article presents a model for handling multi-resolution animation of complex deformable models. An octree-based animation method is described, along with the set of algorithmic tools, that allow for real-time interaction on complex objects. Some efficient collision detection scheme is also described for deformable objects. A method is presented for time-guaranteed mechanical simulation, that allows for automatic tuning of the octree definition to match some external performance constraints, such as adaptation to some pre-defined CPU consumption limitation.

1 Introduction

Real-time physical simulation, especially physical simulation of deformable objects, has been, since about a decade, a very active domain of research, and is still a domain where a lot remains to be done in order to provide real-time simulations involving complex deformable objects. Among the current research open questions is the point to know how to combine the classical need for complexity and model richness, and real-time interaction, which demands fast simulation. In order to provide satisfactory solutions to this problem, many adaptive[1, 2], or multi-resolution [3, 4] solutions have been proposed. For all these sophisticated models, the key point for interactive deformable object simulation is to find the optimal trade-off between real-time computation and simulation complexity. More precisely, this can be seen as the problem to know how much complexity the deformable model can handle, while maintaining the time needed by CPU to achieve the simulation loop (referenced in the remainder of this article as *simulation cost*) under some pre-defined maximal value. Such a compromise, for critical applications domains such as surgical simulation, is probably one of the most difficult compromise to find. Such tuning often goes through time consuming, human made, experiences. This process is all the more rough as changing architecture calculation power completely changes the reachable model definition, for some given average computation time.

This article proposes a multi-resolution deformable model that can provide time-critical computation, i.e. that can automatically adapt its resolution to some pre-defined computation time, that is matched at some given tolerance. This model, based on octree deformable structure, can handle large deformations. Mechanical rules are presented in order to dynamically adapt the model definition, without pre-computation that would limit the maximum resolution the octree can reach. As the mechanical simulation is the main bottleneck in the described algorithm, dynamic mechanical simulation cost measure is used for octree resolution adaptation. This octree is then used both for collision detection (using efficient, stop-at-will algorithm that we will describe below) and mechanical simulation.

The contribution of this article is derived into three parts, each presenting contributions, all together targeted toward the main purpose of this article (time-critical simulation): first is octree mechanical animation, second is collision detection on deformable octree, third is dynamic tuning rules for time-critical simulation.

The remainder of the article is organized as

follows: Section 2 describes related work, referring to adaptive physical animation methods, collision detection process, and time-critical physical simulation. Section 3 presents a novel method for octree mechanical animation, stating the octree configuration is arbitrary, and constant. Section 4 presents a collision technique that can be fit to time-critical context while permitting precise collision detection, once again for a given octree configuration. Section 5 presents time-critical solution for automatic adaptation of the simulation cost, precisely the solution we propose for refining the model definition, or at contrary, to coarsen it, dynamically.

2 Related works

Many research works have been published in the field of adaptive physical simulation. Debunne used octree combined with a mass-spring system, for simulating deformable models in real-time [5], and further presented a solution for introducing more general adaptivity (both time and space [1]) in deformable models. [2] presented a method for adaptive , non-linear finite elements. The CHARMS paper [3] presents a theoretical framework for multi-resolution physical simulation, inspired from subdivision theory, involving higher degree basis function for object representation.

Guaranteed frame-rate has also received quite an attention in recent animation work. [6] introduced the notion of level-of-detail for general purpose animation, potentially allowing for animated model adaptation to guaranteed frame-rate. Few works have been published about guaranteed frame-rate in general-purpose computer graphics. [7] introduced a impostor-based automatic calculation and placement for guaranteed frame-rate during visualization of large scenes. [8] presented a system that dynamically adapts the model resolution to some pre-defined performance, using *cost function* prediction and evaluation. This method has also been improved in [9].

Time-critical constraints is more present in the collision detection literature. In [10] authors introduced a collision detection algorithm that can be stopped at any time. Once interrupted, this algorithm can return an approximated penetrating distance used to prevent the collision. This algorithm is based on the use of a Bounding Volume Hierarchy (BVH) composed of sphere (SphereTree) where each hierarchy level give a better approximation of the object shape while providing a more precise penetrating distance. This work has been extended in [11] and in [12]. Bradshaw [13] explores different SphereTree building algorithm and shows that Medial Axis based algorithm is the most efficient solution for rigid bodies collision detection. He also notice that medial axis algorithm are clearly not design to deformable object simulation and stated that some octree based SphereTree building would be a good candidate to a deformable object critical-time collision system. A recent survey about collision detection for deformable objects [14] points out the fact that efficiency of BVH applied to deformable objects is clearly depending on the BVH update efficiency, during deformation. First attempt to use BVH with deformable objects [15] shows that a bottom-up update lead to better performances than complete top-down hierarchy rebuilding. Yet, even if the bottom-up algorithm performs well in updating a BVH it implies a full hierarchy traversal that is clearly non-optimal. When objects are not in constant collision, too much time has to be spend in just updating the collision model to detect that no collision occurs. In order to avoid the systematic hierarchy update an hybrid algorithm has been purposed in [16]. This hybrid algorithm updates the top-half of the hierarchy in a top-down manner, check collision, and updates the bottom-half of the hierarchy using a bottom-up hierarchy only if a collision occurs. This work showed that deformable collision benefits in using update algorithm that mix collision checks and a local BVH update of the minimal information need to perform the current collision check. In a recent paper, James [17] introduced a new kind of SphereTree dedicated to deformable object, they called BD-Tree. In this BD-Tree each sphere is associated to a reduced number of mechanical samples. From those samples that approximates the deformation field the new sphere position and radius can be efficiently updated in a constant time and independently of sphere location in the hierarchy. These two properties makes a pure top-down al-

gorithm possible with on demand update function mixed within hierarchy traversal.

3 Octree mechanical simulation

This section presents the mechanical method we used for octree animation. Unlike classical use of octree structure, in this article we use octree as a deformable structure. This section first recalls principle of linear finite elements, and then details the mechanical model we propose for simulating octree cell deformations, including details on octree geometry handling, which are necessary for mechanical simulation consistency.

3.1 Linear Finite Element Method

Finite Element Method (FEM for short) provides a continuous deformation field using a decomposition of an object into small elements. This deformation field is linked to internal elastic forces by: $f_{elastic} = f(\Delta x)$, where Δx represents the displacement vector of the nodes (difference of a position x from an initial position x_0). Using first-order approximation of the f function, the previous equation becomes linear: $f_{elastic} = K.\Delta x$, where K is called the stiffness matrix. The stiffness matrix depends on material properties and element geometry. Compared to other FEM approaches, linear FEM is fast (since the stiffness matrix can be precomputed) and stable (because the stiffness matrix is well conditioned). We use hexahedral elements whose stiffness matrix is described in [18] to compute dynamic deformation forces. The system solved at each iteration is [19]:

$$M.\frac{\partial^2 x}{\partial t^2} + C.\frac{\partial x}{\partial t} + K.\Delta x = f_{ext}$$

Despite its stability and swiftness, linear FEM suffers from a lack of precision under large displacements and especially when rotations occur. Indeed, linear elastic forces are not invariant under rotations. Some studies based on corotational formulation (CR) [20] try to extract the rotational part of the movement to keep linear elasticity compatible with large deformations. Two approaches are based on CR : the first works on nodes [19] and the second on elements [21, 22]. Estimating a local, non rotated, coordinate frame for each node does not guarantee that the sum of elastic forces is null (whereas the element based approach does), thus it leads to *ghost forces*. These *ghost forces* provide non-satisfying behavior for free floating objects.

3.2 Octree simulation

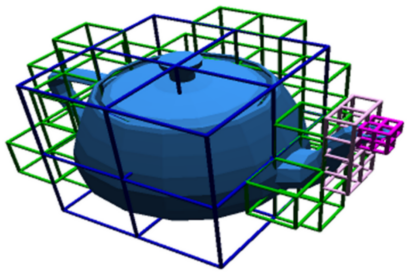


Figure 1: Octree simulation : hexahedral elements of different sizes coexist in the same simulation.

FEM with hexahedral elements usually works on a regular grid. Our aim it to work on a hierarchical structure to locally refine or coarsen some parts of the object. Octree data structure is a common approach for dealing with hierarchical hexahedral elements (see figure 1). Provided with a bounding box of the object we want to simulate, it is quite simple to build the whole hierarchy with a recursive subdivision of an element into eight smaller cubic elements.

However, octree structure classically induces T-junction. T-junction appears when a corner of an element is also strictly inside an edge (or a face) of a bigger element. We call such corner, *virtual node* and the nodes that compose the edge or face *parent nodes*. The figure 2 shows an example of virtual nodes: virtual nodes are circled and white/brighter (resp. blue/darker) nodes are nodes that belong to a face (resp. an edge). These virtual nodes are to be particularized: their movement is not totally free because their position depends on their parent position (see figure 3). Thus, after numerical integration, post-correction is done using projection constraints to insure that *virtual nodes* belong to the right position on the edge (or face).

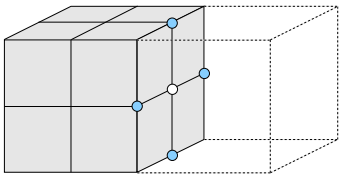
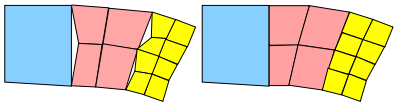


Figure 2: T-junctions: the two kind of virtual nodes are circled.

This approach, by comparison to classical approach for T-nodes (i.e. not simulate them, and use linear interpolation for position calculation), presents the drawback to introduce unnecessary simulation. Yet, first, the simulation on T-knots is fully compensated by the post-correction process, and second, it presents the immediate advantage to make the octree simulation process easy to implement. This solution provides systematic treatment of octree nodes during simulation loop, as well as allowing for stable simulation.



(a) (b)

Figure 3: Constraints on virtual nodes : holes appears without constraints (a) and not with constraints (b) .

3.3 Detecting virtual nodes

The main issue is to quickly detect T-junctions among the elements we simulate. The T-junction detection is in fact about neighbor finding because they appear when there are adjoining elements of different sizes. To find adjoining elements of a given cell, we can neither use hierarchy traversal nor adjacent table because of their cost (time/memory). Schrack [23] proposes an algorithm for same level (= distance to the root) neighbor finding based on Morton Code [24].

The Morton Code [24] consists in a specific indexing of elements from a regular grid. In two dimensions, a element is represented by its coordinates (x, y) computed from a arbitrary chosen corner of the grid. The binary representation of $x = x_{n-1}...x_2x_1x_0$ and $y = y_{n-1}y_2y_1y_0$ are then interleaved to form the Morton code $m = y_{n-1}x_{n-1}...y_1x_1y_0x_0$ (see figure 4 for example). The right neighbor of equal level is obtained by incrementing the x coordinate in the Morton code. The addition starts at x_0 bit and is propagated if a carry is generated. This method induces some problems : carries are propagated one bit at a time and the bits corresponding to the binary representation of y must be passed. In order to overcome such a problem, and make the neighbor isolation closer to simple binary integer addition, some solution have been proposed in the literature. Schrack [23] achieves the neighbor finding in constant time. The propagation of the carries is done in few binary operations. First, the bits corresponding to the y coordinates are saved using a bit-mask. They are then replaced with 1. The addition is then performed and finally the y bits are restored : the result is the Morton code of the right neighbor of equal level. This constant-time search algorithm is trivially extensible to every direction and to higher dimensions (in our case, we obviously use it in 3D).

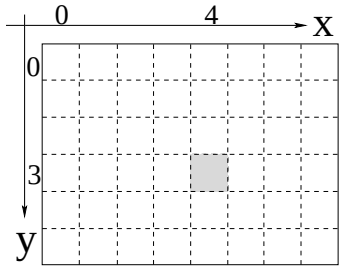


Figure 4: Morton code : coordinates of the darkened cell is (4, 3). The corresponding Morton code is $011010_2 = 26_{10}$.

The weak point of this algorithm is that it only performs well for adjoining elements of same level. As mentioned before, T-junction detection involves different size neighbor algorithm. As a result we would have to complete the neighbor finding using hierarchy traversal. As the cost of hierarchy traversal would be clearly prohibitive in our context, we simplify the T-junction detection problem by imposing the maximal level difference between two adjoining elements to be lower or equal to 1.

This obviously entails some loss of generality for the virtual nodes detection. In practice,

this restriction doest not block the need of multi-resolution in the simulation. Moreover it gives a progressive crossing from coarsened parts of the object to refined parts.

3.4 Optimization : matrix assembly

A common approach in FEM resolution is to loop on every element and to compute elastic forces on the nodes of this element. This prevents from building the global stiffness matrix K (whose dimension is $3n \times 3n$ where n is the number of nodes). However this implies redundant computation : the displacement of a given node will be computed at most eight times (in case of hexahedral elements). A faster operation consists in a loop on nodes instead of elements. For each node i, its elastic forces is given by :

$$f^i_{elastic} = \sum_{j \in nbor(i)} K^j_e . \Delta x_j$$

where K^j_e is an elementary matrix which represents the force contribution of neighbor j on node i, and $nbor(i)$ is the index set of neighbors of vertex i. Obviously, this formulation is equivalent to the resolution of global matrix computation, without building the stiffness matrix.

The two approaches have been implemented. The per node approach is clearly faster than the element based approach : the speed-up measured factor is about 4. However when adding corotational approach, the speed-up decreases and the per node approach is less stable due to ghost forces.

4 Collision detection

In the previous section were presented an octree based framework for adaptive simulation. In this section we will present the corresponding collision model that have the following properties:

- time critical interruptible algorithm;
- hybrid collision detection algorithm that beneficiate from the fact that the object is simulated through a FFD.

4.1 Overview of the Collision Pipeline

Our collision pipeline is composed of three successive steps (fig.5). During the first step (broad phase) the non overlapping objects can be quickly eliminated using the sweep and prune algorithm [25]. During the second step(narrow phase) a SphereTree traversal detects colliding object. During this tree traversal parts of the objects that intersects are located and fires the third step: response computation. This response computation step prevents farther collision using a penalty base method.

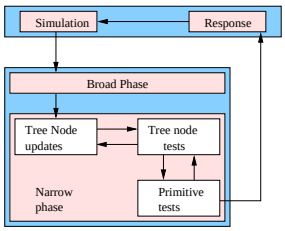


Figure 5: Collision organization within the simulation loop. (from [16])

4.2 Broad phase algorithm

The broad phase algorithm we used is based on SphereTree traversal down to a given depth where a collision response is computed. The depth of the traversal can be adapted intependently of the mechanical resolution through a dedicated Oracle like [10]. The update of spheres position and radius update has been mixed with the checkCollision function in order to only perform the needed updates.

4.3 Update algorithms

We used three differents algorithm to update the sphere positions and radius. Each of them is used to update one subset of the Octree partitioned as the following:

- the AC set: the cells that have been mechanically simulated;
- the UP set: the set of cells upward simulated cells;
- the DOWN set: the set of cells downward simulated cells.

Updating the AC set is straighforward using the simulated cube. Updating the UP set is based on the hybrid algorithm presented in [16]. Finally Updating the DOWN set is based on the

fact that the object's surface is recomputed from the simulated cells through a FFD method [26]. This FFD associate to each points inside of a cell three parameters u, s, t. During the Sphere-Tree traversal each sphere receives a set of u, s, t parameters expressing its position relative to its simulated cell. This simulated cell come from his (grand)-parent and is find during tree traversal. From these parameters sphere position p and radius r can be directly computed as:

$$p = computeFFD(u, s, t, cell)$$

$$r = \text{sphereParent}.r/2$$

4.4 Collision response

A collision response can be computed for any level of the hierarchy using the penalty formula:

$$f = -k * dist$$

where k is the collision stiffness and $dist$ is the penetrating distance measured from the colliding spheres. Once computed the penalty forces have to be transmitted to the height mechanical nodes associated to the simulated sphere. This is done by reusing the three FFD parameters u, s, t of the collinding sphere to weight the force applied to the mechanical nodes.

4.5 Results and Performances

In order to evaluate the robustness of the proposed algorithm, the performances of the presented algorithm has been tested in three simulation situations, ranking from sparse collision context, to a much more densily colliding scene. Fig. 6 counts the number of collisions that have been detected on each sequence. Those test sequences were composed of 500 frames where the presented algorithm is applied. The test has been achieved with different levels of simulated cells. The results are presented in figure 7: in this figure, the snapshot is a ray-traced version from the more complex configuration.

We can notice (fig. 7) that update of the UP set is the main expensive part of the collision detection: on each graph the collision time is clearly linked to the time needed to update the UP set.

	Scene 1	Scene 2	Scene 3
Collis. count	0	11500	1224000
Intersect. test	10500	1598500	4310500

Figure 6: Three collision tests situations with increasing collision count.

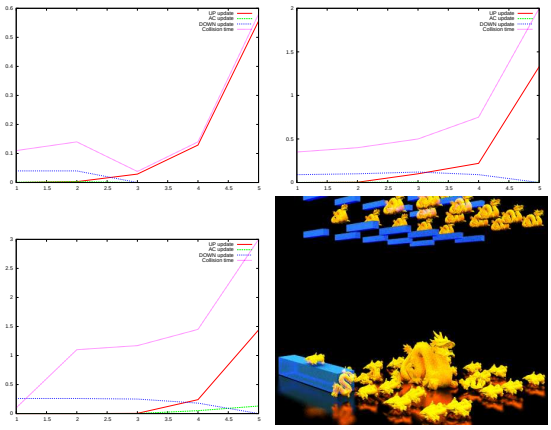


Figure 7: collision performances evaluation (cumulative time in sec). Bottom-right: ray-traced snapshot of the test scene.

5 Time-critical simulation

This section present a method to handle octree resolution changes. During each occurrence of the simulation loop, the algorithm described here is inserted, introducing small changes within octree topology. The overall principle is as follows: some *oracle* predicts, provided with the current simulation cost, and the targeted one, one many topological changes (simplifications, or subdivisions) are needed for the considered simulation step. One the number of simplifications or subdivisions have been determined, the cells to be subdivided/coarsened are then chosen according to the deformation configuration, and the changes are then accom-

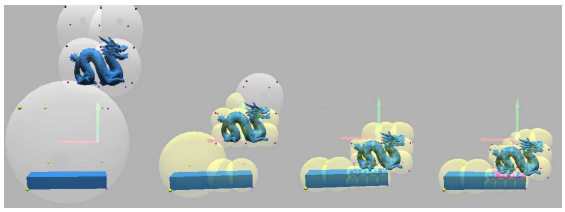


Figure 8: Different step of a falling dragon simulation. Each of the lowest colliding sphere are drawn demonstrating which part of the hierarchy is updated.

plished on the octree structure. In this section, we first present the Oracle algorithm. We then present the method we used for dynamic choice of suitable cells for subdivision/simplification. We finally describe the practical method for handling dynamic changes within the octree.

5.1 Octree resolution changing Oracle

The prediction technique we present here is inspired from the notion of cost function, derived in [8]. The main principle is to iteratively tune the model, so that it asymptotically fits some super-imposed simulation cost.

At the beginning of the simulation, costs for subdividing, as well as simplifying, a cell are estimated, using on-the-fly subdivision and simplification of sample cells, in order to have cost reference on the host computer. Then, during the simulation, when a mechanical iteration is done, the current simulation cost (t_{cur}) is compared to the desired one (t_{des}). If the error, which is $e = \|t_{des} - t_{cur}\|/t_{des}$, is above a given threshold then some multi-resolution operations are allowed. The type of operation (subdivision, or simplification) depends on the sign of (t_{des} - t_{cur}) and the number of operations permitted depends on the difference (t_{des} - t_{cur}) and the estimated cost of a single operation.

Moreover, two sorted queues are built : *mergeQueue* and *subdivideQueue* which contains elements that can be modified. The sort function is based on the laplacian. The elements with high laplacian (resp. low laplacian) are inserted into *subdivideQueue* (resp. *mergeQueue*). Thus high deformed elements are subdivided first as well as low deformed elements are merged first.

To prevent the object from switching too often due to punctual loads of system resources, the cost measure that is used for decision is an average simulation cost, based on a fixed-size history. This average simulation cost is used as t_{des}. Such an averaging process significantly reduces the oscillations during the tuning dynamic process.

5.2 Multiresolution criterion

At each simulation loop, the above-described oracle provides how many subdivisions/simplifications are needed. Once this number of octree resolution changes have been evaluated, elements to be refined or coarsened need to be chosen, in order to maintain, as much as possible, model plausibility. The choice is made accordingly to the fact that octree hierarchy needs to be changed so that constraints on hierarchy level differences on T-junction was respected (see section 3.3). The main (classical) idea is to refine high deformed parts of the object and to simplify low deformed parts. Debunne [1] shows that the laplacian operator gives a good indication of the non-linear feature of the displacement field. Indeed linear variation of the displacement field will be handled correctly because of first-order approximation of linear finite elements. Laplacian is the sum of the second order partial derivatives: close-to-zero laplacian values indicate almost linear deformation field whereas high deformation values show non-linear displacement field and a need of subdivision (see figure 9). Some discrete approximation of laplacian of a function f on a vertex i is computed in in [1] as:

$$(\Delta f)^i = \frac{2}{\sum_{j \in nbor(i)} l^{ij}} \sum_{j \in nbor(i)} \frac{f^j - f^i}{l^{ij}}$$

where l^{ij} is the distance between vertex i and vertex j, and $nbor(i)$ symbolically stands for the integer set of neighbor indexes of vertex i.

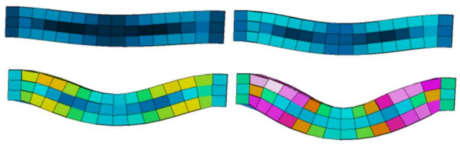


Figure 9: Laplacian as multi-resolution criterion: high laplacian appears on regions with high deformations (thermic scale : colder colors show low values and warmer colors point out higher values).

5.3 Subdivisions and Simplifications

In order to provide efficient handling of topological changes, we need adapted data structure, especially regarding T-structure detection (see section 3). We maintain several arrays during the simulation:

- `activeCellsArray` contains every element which is simulated. This array is used to compute deformation forces when we used element-based approach.
- `activeNodesArray` contains the nodes of the active elements. This array is used to compute external forces such as gravity or collisions and also for deformation forces when we use per node approach.
- `virtualNodeArray` contains nodes from T-junctions. Post correction is applied on nodes that belong to this array.

Subdivision and simplification of elements involves some operations on these arrays. When *transformations* (e.g. subdivisions and/or simplifications) occur, inactive elements and nodes must be removed and new active elements and nodes must be inserted in the two first arrays. New constraints (due to T-junctions) need to be added so T-junctions detection is performed. This detection is local since parts of the objects where there is no transformation will not bring new constraints. For the per node approach (section 3.4), neighbors and their corresponding matrix must be updated for each simulated node.

Some other operations are specific to subdivision or simplification. For example, when a parent element is subdivided into its eight children, some children nodes position need to be updated. This is simply done using linear interpolation of parent nodes position. Furthermore, when an element may be simplified, a test is performed to check that the other seven elements of the parent cell may also be merged.

6 Tests and applications

All the tests and interaction examples have been achieved on a P4 system, using 2.4 GHz CPU, 512 Mo, using Quadro 4 video card. Figure 10 shows a result of an object (the dragon we used for all the snapshots) trying to match some constraints. We imposed, for that test, a desired time of 100 milliseconds and the error allowed is 15%. During the first iterations, the error is very important because we start from an initial configuration where only the root element is simulated and thus many operations are needed to tend to the desired computation time. Then the error is always below the threshold we impose except for some few peaks. These peaks are unavoidable consequence of modern, non real-time, operating systems. As the simulation can only control and measure its own computation time, assumption of low system consumption is always done, which is of course wrong. Yet, as one can see on figure 10, the model globally maintains the pre-required computation cost, providing optimal complexity for the available computation time. Using an average measure of computation time, instead of current measured one, may avoid these peeks, however it would of course introduce damping, and decrease the reactivity of the decision schemes.

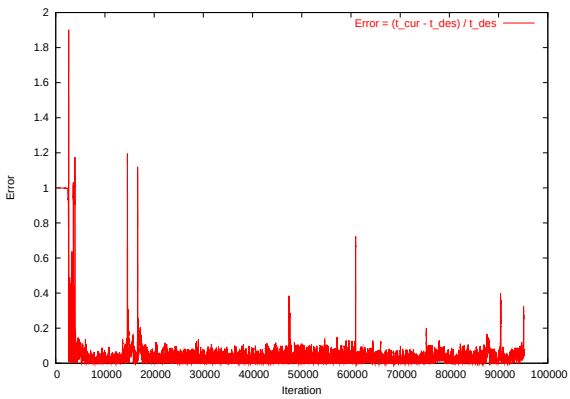


Figure 10: Error measured when imposing time constraint : computation time about 100 milliseconds and 0.15 (15%) of error allowed.

The companion video illustrates the features presented in this article, including the dynamic adaptation to pre-defined computation time. We also included some non real-time simulation, in order to exhibit the fact that our model can also handle stiff systems: in the ”eye-candy” video example, all the small dragons are simulated using stiff, deformable, model.

7 Conclusion

In this article we presented a deformable multi-resolution model that can handle large deformation. It can dynamically adapt its resolution, and can also fit some critical-time computation constraints. Such dynamic adaptation to external

constraint could be strenghtened: our algorithm does not, so far, take into account any perceptual aspects. Taking advantage of parameters such as viewing distance, animation speed (if high enough for any simplification to be worth being achieved), and providing links between classical geometric level-of-detail and physical simulation, in a global tuning algorithm, would be quite challenging.

References

[1] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series. ACM Press / ACM SIGGRAPH, Aug 2001. Proceedings of SIGGRAPH'01.

[2] Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Comput. Graph. Forum*, 20(3), 2001.

[3] Eitan Grinspun, Petr Krysl, and Peter Schröder. Charms: a simple framework for adaptive simulation. *ACM Trans. Graph.*, 21(3):281–290, 2002.

[4] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic. A multiresolution framework for dynamic deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 41–47. ACM Press, 2002.

[5] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Adaptive simulation of soft bodies in real-time. In *Computer Animation 2000, Philadelphia, USA*, pages 133–144, May 2000.

[6] Deborah A. Carlson and Jessica K. Hodgins. Simulation levels of detail for real-time animation. In Wayne A. Davis, Marilyn Mantei, and R. Victor Klassen, editors, *Graphics Interface '97*, pages 1–8. Canadian Human-Computer Communications Society, 1997.

[7] Daniel G. Aliaga and Anselmo Lastra. Automatic image placement to provide a guaranteed frame rate. In *SIGGRAPH*, pages 307–316, 1999.

[8] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH*, pages 247–254, 1993.

[9] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In *SI3D*, pages 95–102, 211, 1995.

[10] Philip M. Hubbard. Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):218–230, 1995.

[11] John Dingliana and Carol O'Sullivan. Graceful degradation of collision handling in physically based animation. *Computer Graphics Forum*, 19(3), 2000.

[12] John Dingliana. *Adaptive Levels of Detail for Interactive Collision Handling*. PhD thesis, Trinity College Dublin, 2003.

[13] G. Bradshaw. *Bounding Volume Hierarchies for Level-of-Detail Collision Handling*. PhD thesis, Trinity College Dublin, 2002.

[14] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, Laks Raghupathi, A. Fuhrmann, Marie-Paule Cani, François Faure, N. Magnetat-Thalmann, and W. Strasser. Collision detection for deformable objects. In *Proc. Eurographics State-of-the-Art Report*, pages 119–139. Eurographics Association, Eurographics Association, 2004.

[15] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, 1997.

[16] T. Larsson and A. Moller. Collision detection for continuously deforming bodies. In *EACG Conference on Eurographics*, 2001.

[17] Doug L. James and Dinesh K. Pai. BD-Tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3), August 2004.

[18] M. Mller, M. Teschner, and M. Gross. Physically-based simulation of objects represented by surface meshes. In *Computer Graphics International (CGI)*, 2004.

[19] M. Mller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *ACM Siggraph/Eurographics Symposium on Computer Animation*, 2002.

[20] C. A. Felippa. A systematic approach to the element-independent corotational dynamics of finite elements. Technical Report CU-CAS-00-03, Center for Aerospace Structure, Colorado, 2000.

[21] M. Mller and M. Gross. Interactive virtual materials. In *Graphics Interface*, pages 239–246, 2004.

[22] O. Etzmuss, M. Keckeisen, and W. Straer. A fast finite element solution for cloth modelling. In *Pacific Graphics*, 2003.

[23] G. Schrack. Finding neighbors of equal size in linear quadtrees and octrees in constant time. In *CVGIP: Image Underst.*, pages 221–230, 1992.

[24] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd. Ottawa, Canada, 1966.

[25] J. Cohen, M.C. Lin, D. Manocha, and M.K. Ponamgi. I-Collide : An interactive and exact collision detection system for large-scale environments. *ACM interactive 3D Graphics Conference*, pages 189–196, 1995.

[26] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH*, pages 151–159, 1986.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Asynchronous Interactive Physical Simulation

Jérémie Dequidt — Laurent Grisoni — Christophe Chaillou

N° 5338

Octobre 2004

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5338--FR+ENG

Asynchronous Interactive Physical Simulation

Jérémie Dequidt , Laurent Grisoni , Christophe Chaillou

Thème COG — Systèmes cognitifs
Projet Alcove

Rapport de recherche n° 5338 — Octobre 2004 — 21 pages

Abstract: This document introduces a multi-agent framework for real-time physical simulation. Unlike classical physical simulators, no shared discrete time-line is imposed to simulated objects. Each simulated object is an autonomous agent that can maintain its simulation state, and possibly modify its behavior by checking its environment. Decision schemes are proposed that enable objects to locally adapt their computation time, in order to maintain approximate global synchronization. Results and measures are presented, especially regarding time management and simulation synchronization in regard of classical simulation methods. Examples are shown, outlining some of the practical advantages such a framework provides.

Key-words: interactive physical simulation, autonomous agents, simulation framework, loose synchronization.

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,
4, rue Jacques Monod, 91893 ORSAY Cedex (France)
Téléphone : +33 1 72 92 59 00 — Télécopie : +33 1 72 92 59 ??

Simulation physique interactive asynchrone

Résumé : Ce document introduit une architecture multi-agents pour la simulation physique temps-réél. Contrairement aux simulateurs physiques classiques, aucune ligne de temps partagée n'est imposée aux objets simulés. Chaque objet simulé est un agent autonome capable de déterminer son état et de modifier son comportement en interrogeant son environnement. Des méthodes de décisions sont proposées pour permettre aux objets d'adapter localement leur temps de calcul afin de maintenir une synchronisation globale. Des résultats et mesures sont présentés, notamment concernant la gestion du temps et la synchronisation de la simulation par rapport aux méthodes de simulation classiques. Des exemples soulignant les avantages pratiques de notre architecture sont donnés.

Mots-clés : simulation physique interactive, agents autonomes, architecture de simulation, synchronisation lâche.

1 Introduction

In modern physical simulations, objects tend to become more and more complex, either saw from the type of geometry they can handle (e.g. cloth [VTJT96, CYMTT92, BFA02], human organs [DDCB01], etc...), or their numerical flexibility (e.g. adaptive simulations [GKS02, CGC+02]). Adaptive objects appear to be very appealing due to their potential ability to change their behavior in order to match some constraints (limited system resources, precision...). However, current simulators are often built for a specific kind of modelling and can hardly handle adaptive or multi-resolution objects. Indeed, building some sophisticated simulation where many objects co-exist, each of them having its own schemes of resolution, is for now pure imagination. We think the main reason of this is the fact that sophisticated simulation often requires ad-hoc context. For example, some mechanical multi-resolution framework results [CGC+02] mention that for the simulation to be adaptive, and real-time to be kept, then implicit differential equation resolution [Cas75] can be used, along with adaptive time-step. This argument is correct, but demands that, if such objects would be integrated along with other objects in a larger simulation, the other objects can handle adaptive time-step too, and are simulated using implicit integration, which is not always natural for very simple simulations. Hence, a simulation involving many objects, among which such adaptive objects, demands specific development and algorithm, even for very simple objects: such a constraint is not natural, and should be able to be overcome.

These past years, within the software development community, Agent-based systems have known a growing success [Woo02]. Such a technique already benefits from extensive success on many Virtual Reality applications [HEV02, Tha00, NNI+03]. Agent based architectures allow for complete encapsulation of treatments and decisions of any kind. Such an approach already gives to virtual humans powerful features, easily providing systems of connection and communication between virtual entities, by providing models for trades, co-actions and collaborations. It nevertheless remains a new research field, and a lot remains to be done before such entity communication can be fully modelled. In particular, the physical simulation field has only been, to our knowledge, poorly studied so far. If such simulation could take advantage of multi-agent based architecture, then all the benefits of agents systems could be included within physical simulators, including the example mentioned, i.e. co-existence on the same simulation of simple objects, and multi-resolution ones.

This paper studies the application of multi-agent principles to physical simulation, and addresses the issues involved. In our model, each simulated object can handle motion resolution, collision detection and control of geometric and mechanical models. Such an object may be easily integrated in a simulation as most of computation is done by the object itself and not by another process. Using this approach, different models may coexist in the same simulation. Because of the autonomy of agents, the simulation is non-synchronous, and no global discrete timeline is any longer available, which is in our opinion the first required step for handling motion adaptivity in complex simulations. Consistency of the simulation is obviously the main question about such a structure: this paper pays special attention to tests and measures regarding simulation synchronization (i.e studies the question to know

if objects, following our scheduling technique, make the simulation consistent, even if no strong synchronization is globally imposed).

The remainder of the paper is organized as follow : section 2 presents some related works on physical simulation, section 3 overviews our framework. Section 4 copes with collision detection for autonomous objects, section 5 deals with autonomous agents. In section 6, tests and results are exposed: measures are provided that show the plausibility of simulation, and few interesting applications are outlined, in order to demonstrate all the potential advantages such a framework provides.

2 Related works

Simulations are usually non-dynamic frameworks: as long as it is intended to develop some complex interaction involving several simulated objects, the physical bodies have common behaviour rules, the integration step is usually the same for every physical body, collision detection uses a common model, etc... Moreover some stages of the simulation loop are, in most architectures, centralized. Thus traditionnal simulation framework cannot be applied with our simulated objects. However the following works propose a more open simulation framework.

In [Mir00], each physical body is a process which receives events for collisions or persistent contacts. Processes have also non synchronized local clock that measures *Local Virtual time*. The synchronization protocol is optimistic and was proved to be correct: the events have a *timestamp* and when a body receives an event, the body is integrated to the timestamp of the event and then the event is processed. Yet, a simulated object A may be rolled back to a previous state if it received events whose timestamp is prior to the body local virtual time. Other objects that are interacting with A received events to let them know they have to roll-back. Due to the roll-back mechanism, this may not be used for interactive simulation.

OpenMASK [Mar01] is a synchronous distributed simulation in which animated objects can work at different frequencies. The approach is mainly parallel: a centralized scheduler computes the simulation frequency, keeps synchronization between the objects and schedules tasks on all hosts. Such a framework is also not fully designed for physical simulation.

The BREVE system [Kle02] provides multi-agent framework for non-centralized systems, possibly handling physical simulation. In the breve system, the collision process involves global, classically centralized process: possible collision entails event production, hence providing immediate treatment of the response. This closely relates to classical, centralized, simulation. By comparison, we tried to relax such a condition, by providing each agent the possibility to decide of its own simulation rythm, and define logical adaptation rules, included to each agent.

Chenney [Che99] proposed an event-based simulation on rigid bodies, in order to combine both the need for simulation consistency, and the need to compute only needed simulations. Such is only applicable to rigid bodies, as deformable bodies usually demand more integration computation for plausible simulation. In addition to that, the work presented in this paper, by contrast, tries to separate simulation time from actually detected collision

time. Simulation frequencies are changed by the agents themselves, according to respective delay of simulation between agents, instead of using event-based programmation, that would super-impose computation frequencies to agents.

Some other works study problems related to the issue we address here. [BW97] discusses the possibility, for heterogenous simulation systems, to collaborate in order to achieve one global simulation. Constraints handling between systems are discussed. Our approach goes one step beyond this work, by setting each objet (not sets of objects of the same simulation kind) as an autonomous simulation, without any global discrete time-line superimposed to objects (by comparison [BW97] provides a common discrete time-line for all the simulations).

3 Technique overview

Our framework may be seen as a multi-agent system: each dynamic object is an agent that, given some internal data, computes its next state. In classical simulation, forces and torques are computed, then the motion is resolved, using some integration numerical tool, to provide new state of the system: spatial position and orientation. Communication between entities are restricted to collisions and constraints as they can change the motion of an entity (because they create new forces and torques). In the case of rigid bodies [Mir00], only a force and a torque is required. However for deformable bodies, many forces may be created due to collisions.

To ensure correctness of interactions and realism of simulation (further called *simulation consistency*), bodies must be at *somewhat* the same simulation time. Indeed a collision detected between 2 bodies A and B, which are too distant in the simulation timeline, is non-sense. Explicit synchronization is often used to grant simulation consistency: after each dynamic object has done its iteration, data are collected by another process to compute forces due to interactions and so on, this leads to synchronous simulation. However explicit synchronization does not fit with the independency of the multi-agent system paradigm. Such an independency has two main consequences:

- First, the collision detection framework should be designed to work with autonomous objects. Actually most of existing architectures involve a global treatment, which does not go in the direction of object independence.
- Second, one needs to make sure the simulation remains consistent, even when the synchronization is relaxed, and no strong synchronization is superimposed to objects: every simulated object should be more or less at the same simulation time. Autonomous agents take into account some information about the environment to give consistency to the simulation. They can dynamically decide to increase / decrease their simulation or modify their resolution scheme whether they are to slow or not.

In the following sections we detail these two issues in detail. In section 4, we present a collision detection framework that insure autonomy of the objects and in section 5 the global framework is exposed explaining how the simulation works and how it remains consistent.

4 Collision detection aspects

Our collision detection and treatment to collisions should insure, as much as possible, the autonomy of simulated objects. It means that a simulated body has to detect its collisions with other bodies and has to adapt its dynamics accordingly. Moreover, the collision and treatment process should be effective so as to work in an interactive simulation.

Given two colliding objects and an estimation of their intersection, penalty forces is a simple way to separate the objects: a force in proportion with the intersection estimation is computed according to the spring-damper model. The estimation of intersection may be the penetration depth which is the shortest vector that bring two colliding objects in touching contact.

Therefore, our goal is to propose a framework which detects collisions, computes penetration depth when collisions are detected and both of these stages should be done by simulated objects.

4.1 Rigid convex polyhedra

Lots of works have been produced on collision detection for rigid convex polyhedra (see [LG98, JTT01] for surveys). Algorithms based on the *GJK* [GJK88] are among the fastest and most robust algorithms. *ISA-GJK* [Ber98] is an incremental algorithm designed for convex polyhedra that quickly determine if two polyhedra are colliding or not (boolean answer). The computation of force response has, so far, been a bit less studied to our knowledge. Yet, the use of penetration depth for force response evaluation has so far proved to be quite a good compromise [KOLM02, KLM02]. We choose to use the *Expanding Polytope Algorithm* [Ber01], also based on the GJK, that uses *ISA-GJK* as a start point to get the penetration depth between two polyhedra.

These two stages can be done for autonomous objects: knowing the position of all the polyhedra in the scene, an object can check the collision and compute penetration depth if necessary. Moreover using two GJK-algorithms is no concurrence: initialization of the *Expanding Polytope Algorithm* is done by the last iteration of *ISA-GJK*, reducing the total computation time.

4.2 Deformable Objects

Collision resolution is far more expensive for deformable models, thus most frameworks use approximate methods to estimate penetration depth. Many recent works are available on deformable models [LM01], but using spheres for collision on deformable bodies is interesting [DJK97] because computing the penetration depth between two colliding spheres or between a sphere and a polyhedron is simple: for two colliding spheres, the penetration depth is $r_1 + r_2 - d$ where r_i are the radius and d the distance between the radii of the two spheres. Moreover, the Expanding Polytope Algorithm allows to determine penetration depth between a sphere colliding a polyhedron. Spheres clearly have the drawback to allow only for poor collision precision, especially regarding deformable objects such as those used

in section 6 (i.e. tissue): yet, they provide very fast collision tests, and can be adapted at will for precision increase [GNRZ02].

The main issue is to handle collisions between a high number of colliding spheres: consider a single sphere A that potentially collides a deformable object B (where B is composed of hundreds of spheres). It is obviously time-consuming to check collisions between A and each sphere that composes B, whereas most of spheres of B are far from A.

To accelerate the collision process, an internal voxel grid is built [DJK97] (figure 1). Each rectangular voxel contains spheres from the decomposition. So to test collision between A and B, A is placed in the voxel grid and collision are computed between A and the spheres that belong to the voxels in which A takes place (darkened voxels). Thus only spheres that are close to A need collision detection (blackened spheres).

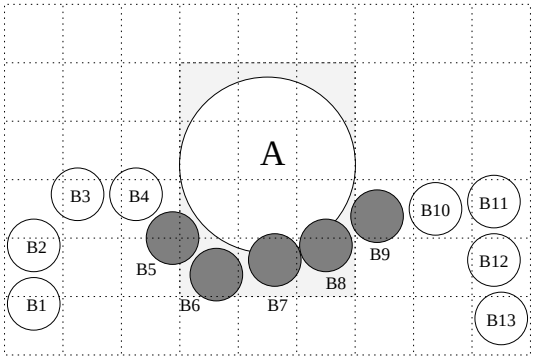


Figure 1: Internal voxel grid for deformable objects : deformable object B is in collision with the sphere A. Collisions are computed only between A and the blackened spheres.

4.3 Common pipeline

It is time-consuming for an object to know the position and to check the collision of all the other objects. Thus we build a pipeline [Zac01] for collision detection, in order to diminish its cost. We divided the scene space into regular zones in which a *zone agent* is defined. Agents get bounding volumes of every object in the zone and apply *Sweep And Prune* algorithm [CLMP95] to quickly determine potential colliding pairs. When a potential collision is found, the agent informs both objects that are concerned and then they send their position to each other and then apply the final stage (*ISA-GJK* and *Expanding Polytope Algorithm* for polyhedra, or algorithm based on sphere decomposition for deformable models, using voxel grid optimization mentionned above). We choose to use k-DOP [Zac98, KHM+98] as bounding volumes : they are a good approximation for most objects and are easy to construct and to update due to *GJK*.

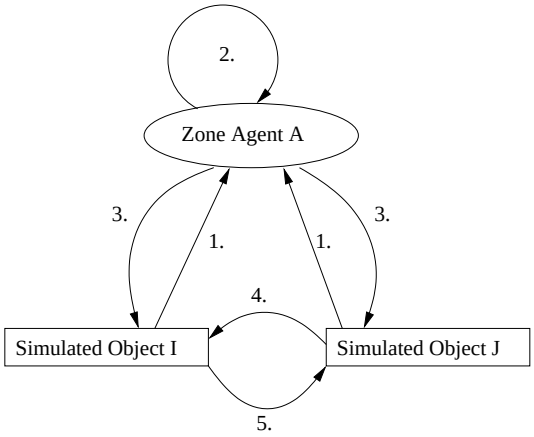


Figure 2: Example of two colliding polyhedra (I and J) in the zone A.

To make things clearer, consider figure 2. First each polyhedron send its k-DOP to a zone agent (arrows 1). Then the zone agent finds potential colliding pairs (arrow 2) and informs each polyhedron of its potential collisions (arrows 3). As I and J may probably collide, J sends its position to I (arrow 4). I detects a collision with J,then computes the penetration depth and sends it to J (arrow 5).

All of these added stages reduce the computational cost of collision detection in order to use it in interactive simulations. Moreover most of computations (especially costly computations) are done by the simulated objects, insuring autonomy for the agents.

4.4 Discussion

The collision approach presented in this section aimed at providing simulated agents with collision process that would be as "agent-like" as possible. In that way, we tried to minimize the global processing part of the collision detection, and optimize the collision pipeline. It is to note that, clearly, such an approach has the main properties:

- First, classical collision problems of multiple contacts between rigid objects are not addressed by our approach, as they are out of the scope of the problem addressed in this article. About this problem, the approach presented here does not bring anything new on this, as it is based on existing results about geometrical collision detection. Techniques like global optimizations [MS01], especially the question to know is such approaches could be linked to agent-based techniques for simulation, are indeed interesting, but are far beyond the scope of this paper.

- We use penalty method for force response calculus: as a result, we do not try to handle exact contact between objects: this presents the immediate drawback that simulated

objects can potentially geometrically intersect each other during the simulation, but also has the advantage that the relaxed synchronization of the simulation can actually be done.

Finally, it should also be noted that the notion of zone agent mentioned in this section does not refer to the same type of *agent* than in the other sections of this article: we named this likewise in order to remain within the same terminology. Yet, they can be seen as practical ways, or available tools for animated agents, for easily finding the objects potentially colliding them. Zone agents are not scheduled (see section 5), like other agents composing the framework.

5 Non-synchronous simulation time handling

As explained in section 3, our framework is an agent-like system that needs consistency to make the global simulation realistic. Consistency cannot be achieved with explicit synchronization within our context. Instead of imposing constraints to agents, we choose to control the behavior of an agent by defining local decision rules, that let each agent acts on its simulation. In order to produce efficient, and reactive enough system, we chose to built our own scheduler that coordinates tasks (dynamic objects). This scheduler (that is build in a quite classical manner) insures autonomy of agents: when an agent is provided with execution time, the agent animation loop iteration is called, and such a call is not interrupted until the function call terminates. Within this call, each agent has the ability first to calculate its next simulation state, but also to send to the scheduler its priority change wish.

5.1 The scheduler

Our scheduler is a priority based, FIFO scheduler that schedules tasks (i.e. autonomous agents) based on their priorities. A set of $N+1$ integer priorities is defined within the simulation: each agent a is assigned some initial priority $p(a) \in \{0, 1, \cdots, N\}$ (high priority value corresponds, in our notation, to intensive computation objects). The way our scheduler works is close from classical system scheduler: a queue Q_i is associated with each priority i, a task a is included in queue $Q_{p(a)}$. Each time an agent is activated, it can dynamically change its priority level, in which case it is then queued in its new priority queue (see section 5.2.1 for details). One additional array, S, stores objects that temporarily dropped off and do not need system resources for a moment (see section 5.2.2 for details about this). Each queue $Q_i,\ i \in \{0, \cdots, N\}$ is associated with an execution integer n_i, that can never exceed some pre-defined value max_exec: the n_i are used to control the execution frequency, and avoid execution starving problems. In pseudo-code description, the scheduler algorithm could be described as follows:

$init(S, Q_0, \cdots, Q_N)$;
$prioCurrent{=}N$;

do forever:
  $n_{prioCurrent} = n_{prioCurrent} + 1$;
  if ($n_{prioCurrent}$==max_exec) then
    $n_{prioCurrent} = 0$;
    $prioCurrent = (prioCurrent - 1) mod (N + 1)$;
  else
    a=*getQueueHead*($Q_{prioCurrent}$);
    callAnimationStep(a);
    queue(a, $Q_{p(a)}$);
    $prioCurrent = N$;

It is important to understand that *callAnimationStep* on agent a can change its priority, according to rules defined in section 5.2.1, and hence, that the queue storage might not store agent a in the queue it was initially inside. The *queue* function can either store agent a in its new treatment queue, or store it in array S if necessary (see section 5.2.2). The proposed scheduling algorithm, simple to implement, ensures that agent of priority k are statistically activated for animation max_exec times more often than agents of priority $k - 1$: as a result, frequency call, as a function of priority, is an exponential rule.

Tasks stored in S are stored along with a requested wake-up time: S is constantly sorted according to this wake-up time, in order to keep efficient treatment. Agents are woken up using event propagation are reinserted in ad-hoc priority queue when requested. Each autonomous agent may completely stop its simulation for a period of time with this queue: yet, an agent never stops taking decisions such as priority change, or sleep time duration modification. This ensures that no artificial, non-recoverable decay would be inserted within simulation, without knowledge of the concerned agents.

This scheduler is designed so that a consistent simulation can only be achieved with the cooperation of every agent, so the main thing for the agent is that all the dynamic objects controlled by the agents should be at the same simulation time (this is the consistency criterion for the global simulation): each agent has, apart from the task to achieve its own simulation, to dynamically adapt its computation intensity to the global simulation speed. The following subsection describes the proposed technique.

5.2 Simulation control

All the control is given to agents themselves about their simulation. As explained above, the scheduler allows each agent to change the way it is computed, through two means: priority handling and sleep. In order to grant simulation consistency, efficient decision schemes within agents are needed on these two aspects, for defining the behavior of an agent. For such a purpose, an agent obviously needs to get information about its environment. As a result, the *callAnimationStep* function, used in the above algorithm, also transmits some information to agent a: the simulation time of the fastest agent and the simulation time of the slowest one. It is important to note that, as no common discrete time-line is imposed to objects, the only common time-line is the continuous simulation time. Depending on what

we will call from now on the current *simulation time window*, rules are used by agents for priority change decision, or, in case of important decay, sleep for a given simulation duration.

5.2.1 Priority modification

The algorithm used for priority modification is very simple: when an agent is on minimal (respectively maximal) time simulation (given by the scheduler at activation), agent priority is increased by 1 (respectively decreased by 1), for next activation, taken into account by the scheduler according to algorithm describe in section 5.1. By dynamically changing priorities, the gap between the fastest agent and the slowest one tends to decrease. As every agent acts in the same manner, agents tend to converge to the same simulation "speed" (on the simulation time-line). In order to make the system flexible, little delay between the simulation time of the fastest agent and the slowest one is permitted. Actually, temporal simulation coherency has among other consequences that, when simulation time distance between two objects interacting together is small, then the introduced errors, in comparison to classical, synchronized simulation, are negligible, provided that movements are of reasonable magnitude. Agents do not modify their priorities unless the delay between the extremal agents is greater than a pre-set *threshold*: this way, priorities changes are only achieved when needed.

5.2.2 Simulation sleep

Another way for an agent to control its simulation is to interrupt simulation computation for a while, by asking the scheduler to be called back only after a given amount of time. The idea is to dynamically evaluate how much CPU time a simulated object needs. The principle of this scheme is actually pretty simple: evaluation of simulation speed, simply by comparing evolution of simulation time, related to simulation window (i.e. maximal and minimal simulation time for all agents), easily provides some evaluation factor of CPU consumption accuracy for the considered agent, relatively to others. Once an object has evaluated the computation time it needs and placed it in regard to other simulated objects, it can interrupt itself after an iteration (and ask the scheduler not to be called before some given simulation time), allowing the other objects to do their own iteration. Then when it resumes, all objects are supposedly at the same simulation time, which ensures consistency of the global simulation.

modifyBehavior(curTime,minTime,maxTime)
$midTime$=($maxTime$+$minTime$)/2
$delta$=($midTime$-$locTime$)/($maxTime$-$minTime$)
$ratio$=$ratio$+$c1$*$delta$
$wake$=$curTime$+$lastCompTime$*(1-$ratio$)/$ratio$

$ratio$ is the fraction of CPU time the object is ideally consuming. Computation time of an object is bound to be irregular throughout time, because of collision computation, or numerical integration (implicit integration requires sometimes several iterations). Ac-

cording to the new *ratio*, it delays its next simulation call, allowing other objects to do their simulation: In the above algorithm description, $curTime$ represents the time (continuous simulation time) when the scheduler calls the function $modifiyBehavior$, $wake$ is the estimated time when the object will wake up and $locTime$, $minTime$, $maxTime$ are respectively the local simulation time, the minimal simulation time, the maximal simulation time. The modification of *ratio* is based on a value *delta* and a constant $c1$. The *delta* is a signed value included in $[-1/2, 1/2]$ which represents the distance between an extremal simulation time and the center of the simulation window. The closer $locTime$ is to the middle simulation time, the closer will be *delta* to 0, which will lead to a slight modification of *ratio*. Conversely, if $locTime$ is close to $maxTime$ (resp. $minTime$) then *delta* will be close to $-1/2$ (resp. $1/2$), which will create an important decrease (resp. increase) of *ratio*.

In other words, simulation interruption artificially delays computation of some objects. It can work in combination with, or independently from priority modification, in order to bring consistency to the simulation. When combined with priority modification functionality, simulation interruption adds another mean for simulated objects to control their behavior (for example by trying to respect some pre-set simulation frequency).

5.3 Comparison to centralized simulation system

Apart from the question to know if objects can actually remain, under some given tolerance, all together at the same simulation time, the most sensible question is the cost overhead point: actually, the scheduling task does not appear in some classical simulation architecture, and potentially demands additional computation time, while classical simulation is already quite computationally expensive. The task switching cost can be ignored in our test platform, as it reduces to simple function call (no multi-thread or multi-process structure is used). Looking at the scheduling algorithm, one can see that no complex computation is involved, and the computation overhead is only $O(1)$ per simulated agent.

Another point is that our approach has the important advantage that it can quite easily take advantage of multiple-CPU architecture, without specific additional efforts. As all memory writings are only achieved by the agent the data belong to, the potential concurrent access on data would be read-only, and the management of such access could be reduced to its most simple expression. By comparison, classical simulation, in order to provide the same properties, would need to get to more classical parallel numerical algorithms, which would demand a full re-writing of the resolution kernel.

6 Tests and applications

First, we present some tests aiming at measuring the efficiency of our framework : especially tests measuring the consistency of the simulation, i.e. the simulation time possible decay between objects. Then, as our agent-like framework offers the possibility to mix simulated objects of different kind, we describe practical examples that illustrate the potential benefits physical simulation can take from agent-based approaches.

6.1 Framework Validation

Framework validation, and/or comparison to classical, synchronous simulation, is indeed a difficult question. The tests we achieved try to show that in some "simple" case, our framework behaves in the same manner a classical global simulation would do. To test our framework, we have run simulations on Intel PIV 2.6 Ghz with 512 MB Ram. The test scene is composed of about ten non-deformable spheres falling on a plane. The integration step is constant, and set to 1 millisecond, using explicit Euler integration. Each simulated object has, at the beginning of the simulation, a random priority.

The main thing in our framework is to insure simulation consistency: this is linked to the delay between the fastest object and the slowest one. In our test scene, we measured the delay between the maximal simulation time (*fastest* object) and the minimal simulation time (*slowest* object). Figure 3 exposes the results. 10 000 instant measures where taken during simulation and all delays were lower than 16 msec. About 10% of the samples are below 6 msec, and most samples are between 8 msec and 11 msec which is confirmed by the average delay : 8,5 ms.

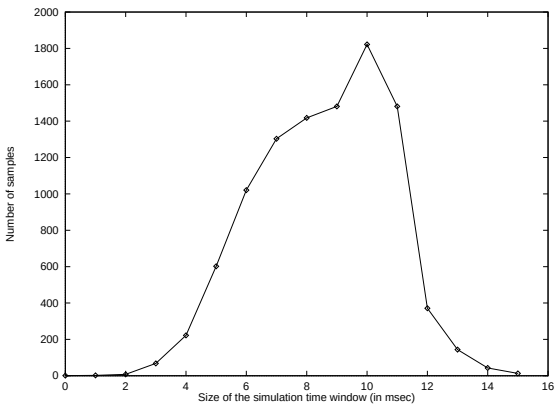


Figure 3: Repartition of delays between the maximal and minimal simulation time.

The average simulation time delay is about 8 milliseconds when we consider simulated objects whose integration step is 1 millisecond. Thus for the objects with extremal simulation time, the delay is about 8 simulation iteration. Because of temporal coherency that is, most of the time, available in interactive simulations, we can consider that the delay is small enough for providing a realistic simulation. Moreover, as we only record the delay between the fastest object and the slowest one, we assume that the delays between common objects (e.g. apart from the two extremal objects) are smaller.

Another possible test is to monitor the priority evolution during a simulation. We consider the same test application as previous, where the initial priority is randomly set for each simulated object. As all the objects are of the same nature, they should have more or

less the same computation time, and one would expect them to tend to the same priority. We snapshot objects priority at regular temporal step and put the results on figure 4. The curves have been smoothed for clarity, which leads to non-integer priorities. Actually, all the objects priority tend to converge to the same value, which matches the intuitive behavior one would expect from each autonomous agent.

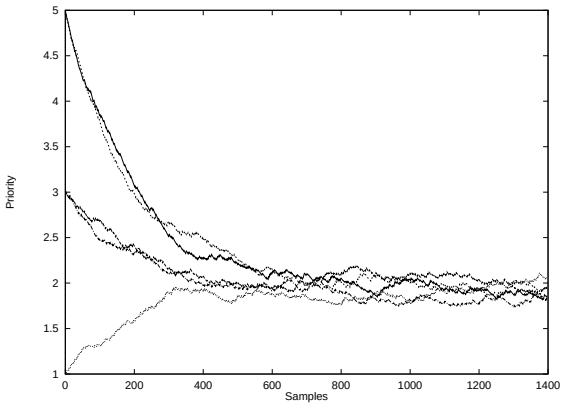


Figure 4: Priority evolution for 5 simulated objects (curves are smoothed for better readability, which creates non-integer priorities).

6.2 Sample applications

Apart from framework validation, it is natural to expect from agent-like framework to provide the same result as classical simulation frameworks, but also to propose extensions to it, in regard of simulation possibilities. We first present two example applications, that show that non-synchronous framework provides the same kind of functionalities than classical ones: first, complex simulation on rigid, convex bodies, and second, interaction on deformable model. We also provide here examples that show agent-based structure flexibility and advantages: first, simulation involving different integration schemes for objects (without loss of generality, mix between explicit and implicit numerical integration is provided as an example). Second, we provide example of complex interaction on several deformable models, where advantage is taken of implicit integration numerical stability in order to adapt time-step of each deformable model, depending on its interaction context.

6.2.1 Rigid convex objects simulation

The simplest application of our framework is rigid body simulation: no complex numerical integration is needed nor collision computation (our objects are spheres). Figure 5 exposes hundreds of rigid spheres falling on fixed bricks.

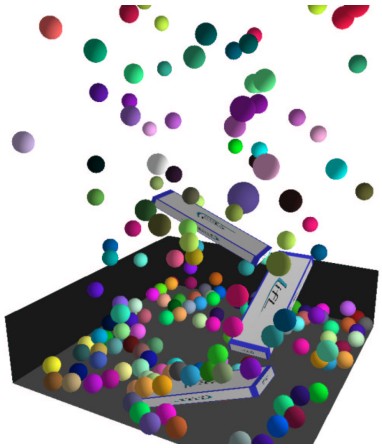


Figure 5: Falling spheres on fixed bricks.

6.2.2 Interaction on deformable objects

This example (figure 6) combines a rigid object and a deformable one. A deformable cloth is hanged by its four corners and a solid sphere, manipulated by the user, interacts with the cloth.

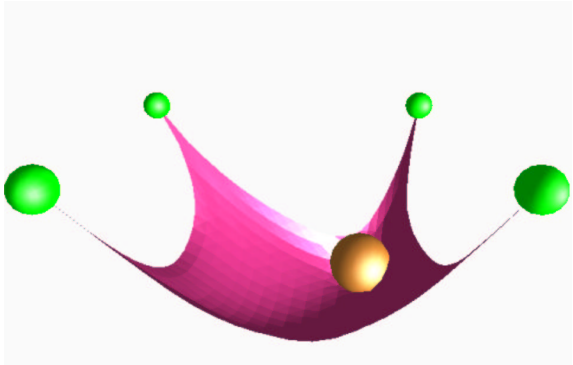


Figure 6: A Solid sphere interacting with a deformable cloth.

6.2.3 Heterogeneous integration

Because of the agent-based structure, no condition on the used integration numerical technique is super-imposed. It is actually straightforward, using the framework presented here, to combine objects with explicit integration (which is known to provide non robust, yet very fast, integration that can provide satisfying results on simple PDE) and objects with implicit integration (more robust, but more complex integration process), which allows to take benefits of both schemes. Figure 7 includes rigid spheres which use Euler explicit integration (for rapidity) and a deformable cloth which uses Euler implicit integration (for stability and precision). Thus computation time is not wasted to compute the movement of rigid spheres (as explicit integration is not too time-consuming). A companion video illustrates such property: a set of spheres (that all use simple explicit integration scheme) fall under the action of gravity within a tissue (that uses implicit integration, for better realism and stability). These spheres are then interactively manipulated by user, using some interaction sphere.

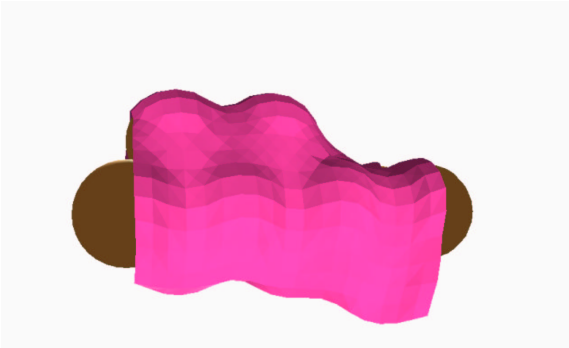


Figure 7: Example of heterogeneous simulation mixing explicit (on tissue) and implicit integration (on rigid spheres).

6.2.4 Adaptive time-step

As our framework provides consistency and important freedom for simulated objects design, one can control the simulation more precisely. When an object is close to stability (e.g. its movement does not change significantly), it can reduce its computation time: a possible way to achieve this is to modify the integration step dynamically. Some works [BW98] introduce adaptive integration step for implicit integration to meet some stability criteria. The method we used is the following: when the percentage of movement (i.e. euclidean distance between two consecutive state vectors for the considered object) for an object is below a fixed threshold for some time, this means that the object is close to stability. As a result, integration step may be increased, in order to save computation time (or possibly,

simulation may even be stopped). Conversely, interaction on the object, or collision with some other object entails automatic and event-based reconfiguration of the sleeping agent for reset of simulation time-step, as well as sleep time shortening (i.e. sleep time changing, see section 5). Figure 8 shows an example of such application, where 3 clothes are interactively

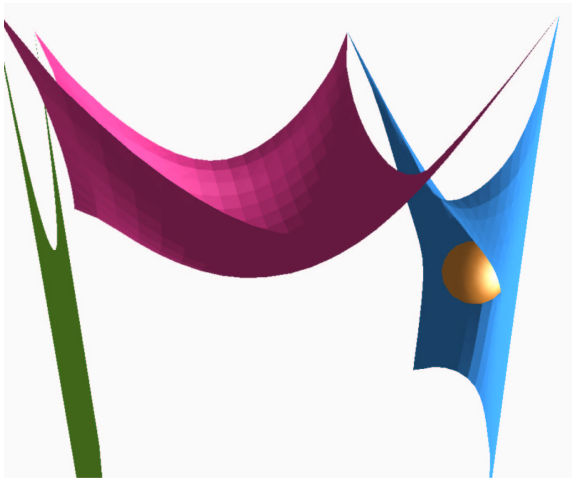


Figure 8: Application with adaptive time-step: the manipulated cloth has a smaller integration step than the other clothes that are in a stable state.

manipulated by user. A companion video illustrates real-time interaction on such models: on this video, where a set of 3 tissue is manipulated in real-time by user using some simple sphere, one can see that each tissue reacts in a satisfying manner, and that tissue interaction can be done at will by user, in spite of possible time-step modifications. This simulation takes full advantage of the fact that during the interaction, always at least two of the clothes are left hanging, while the third one is being manipulated. Timestep adaptivity allows for optimal CPU use, and better interaction than non-adaptive integration: in the non-adaptive case, too small a timestep makes a slow simulation, and too big a timestep provides poor numerical accuracy and weakens interaction with a model. Adaptive timestep simulation, where timestep is adapted for each object would be quite tricky to provide in a global simulation and is, by comparison, very straightforward to design in the agent-based framework presented here.

We think that the main advantage of such approach lies within the independence that each agent is provided with. The ability, for some object, to dynamically adapt its resolution time-step has no consequences on the algorithms involved for the other objects. On the simple illustrated example of Figure 8, it is clear that some centralized system, provided with explicit synchronization and timestep adaptivity, would somewhat provide the same features. Such centralized framework would demand complex numerical handling of the

global simulation resolution system, and ad-hoc data structure for such. By comparison, in the presented approach, the features added to some object simulation have no consequences at all on the design complexity of the other involved objects: each object remains responsible of its own complexity, both in terms of algorithm cost, and software design. Adaptive timestep is only one practical advantage that simulated objects can benefit of, within agent-based approach: no kernel adaptation was needed for our test, neither was it for use of heterogeneous integration techniques.

7 Conclusion

Our framework provides multi-agent system-like for physical simulation. Rigid bodies, as well as deformable objects, can be handled. This framework allows for hybrid method of integration, and may potentially be used on quite different types of objects (adaptive ones or very simple model). Most steps of the simulation are done by the agents themselves to insure autonomy and independency.

A lot remains to be done before all the advantages of agent-based framework for simulation would be completely perceived. Constraints between objects especially become a sensitive point in this framework: is a multi-body [Mir00] still a set of objects? What should be done, and how, when a set of objects is dynamically linked in order to create a multi-body?

We think this framework would be a good tool for complex, adaptive, physical interactive simulation, where many objects would interact together, each having its own sophisticated physical model. Adaptive time-step for implicit integration within a complex environment is only a small step toward really flexible practical simulation; providing simulators with stable dynamic regulation rules, that would automatically adapt computation cost to interaction situation, would also be of high usefulness.

References

[Ber98] G. Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphic Tools*, 4(2):7–25, 1998.

[Ber01] G. Bergen. Proximity queries and penetration depth computation on 3D game object. *Game Developers Conference*, 2001.

[BFA02] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *Proc. Siggraph 2002*, pages 594–603, 2002.

[BW97] D. Baraff and A. Witkin. Partitioned dynamics. Tech. Report CMU-RI-TR-97-33, The Robotics Institute, Canregie Mellon University, 1997.

[BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH'98*, pages 43–54, 1998.

[Cas75] J. R. Cash. A class of implicit runge-kutta methods for the numerical integration of stiff ordinary differential equations. *Journal of the ACM (JACM)*, 22(4):504–511, 1975.

[CGC+02] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic. A multiresolution framework for dynamic deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 41–47. ACM Press, 2002.

[Che99] S. Chenney. Asynchronous, adaptive, rigid body simulation. In *Proc. SIGGRAPH 99, technical abstracts and applications*, July 1999.

[CLMP95] J. Cohen, M.C. Lin, D. Manocha, and M.K. Ponamgi. I-Collide : An interactive and exact collision detection system for large-scale environments. *ACM interactive 3D Graphics Conference*, pages 189–196, 1995.

[CYMTT92] M. Carignan, Y. Yang, N. Magnenat-Thalmann, and D. Thalmann. Dressing animated synthetic actors with complex clothes. *Proc. Siggraph 1992, Computer Graphics*, 26(2):99–104, 1992.

[DDCB01] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series. ACM Press / ACM SIGGRAPH, Aug 2001. Proceedings of SIGGRAPH'01.

[DJK97] Sung Yong Shin Dong Jin Kim, Leonidas J. Guibas. Fast collision detection among multiple moving spheres. In *13th annual symposium on Computationnal geometry*, August 1997.

[GJK88] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, RA-4:193–203, 1988.

[GKS02] E. Grinspun, P. Krysl, and P. Schröder. Charms: a simple framework for adaptive simulation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290. ACM Press, 2002.

[GNRZ02] L. Guibas, A. Nguyen, D. Russel, and L. Zhang. Collision detection for deforming necklaces. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 33–42. ACM Press, 2002.

[HEV02] Z. Huang, A. Eliëns, and C. Visser. 3d agent-based virtual communities. In *Proceeding of the seventh international conference on 3D Web technology*, pages 137–143. ACM Press, 2002.

[JTT01] P. Jiménez, F. Thomas, and C. Torras. 3D collision detection: A survey. In *Computer Graphics*, volume 25, pages 269–285, 2001.

[KHM+98] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *T-VCG(4)*, pages 21–36, 1998.

[Kle02] J. Klein. breve: a 3d environment for the simulation of decentralized systems and artificial life. In *Proc. of 8th international conference on artificial life*, December 2002.

[KLM02] Y. Kim, M. Lin, and D. Manocha. Fast penetration depth estimation using rasterization hardware and hierarchical refinement. In *Workshop on Algorithmic Foundations of Robotics*, December 2002.

[KOLM02] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *ACM Symposium on Computer Animation*, July 2002.

[LG98] M.C. Lin and S. Gottschalk. Collision detection between geometric models : A survey. In *IMA Conference on Mathematics of Surfaces*, 1998.

[LM01] T. Larsson and A. Moller. Collision detection for continuously deforming bodies. In *EACG Conference on Eurographics*, 2001.

[Mar01] D. Margery. *Environnement logiciel temps-réel distribué pour la simulation sur réseau de PC*. PhD thesis, Université de Rennes 1, 2001.

[Mir00] B. Mirtich. Timewarp Rigid Body Simulation. *ACM SIGGRAPH*, pages 193–200, July 2000.

[MS01] V. J. Milenkovic and H. Schmidt. Optimization-based animation. In *SIGGRAPH*, pages 37–46, 2001.

[NNI+03] H. Nakanishi, S. Nakazawa, T. Ishida, K. Takanashi, and K. Isbister. Can software agents influence human relations?: balance theory in agent-mediated communities. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 717–724. ACM Press, 2003.

[Tha00] D. Thalmann. The virtual human as a multimodal interface. In *Proceedings of the working conference on Advanced visual interfaces*, pages 14–20. ACM Press, 2000.

[VTJT96] P. Volino, N. Magnenat Thalmann, S. Jianhua, and D. Thalmann. An evolving system for simulating clothes on virtual actors. *IEEE Computer Graphics and Applications*, 16(5):42–51, 1996.

[Woo02] M. Wooldridge. *An Introduction to Multiagent Systems.* Wiley & Sons, 2002.

[Zac98] G. Zachmann. Rapid collision detection by dynamically aligned DOp-trees. *Proceedings of IEEE, VRAIS*, 1998.

[Zac01] G. Zachmann. Optimizing the collision detection pipeline. *First International Game Technology Conference (GTEC)*, 2001.

Unité de recherche INRIA Futurs
Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399