École Polytechnique **École Doctorale**

Simulation de chirurgie par cœlioscopie : contributions à l'étude de la découpe volumique, au retour d'effort et à la modélisation des vaisseaux sanguins

THÈSE

présentée et soutenue publiquement le 19 Mai 2003

pour l'obtention du

Doctorat de l'École Polytechnique de Paris

(spécialité Mathématiques et Applications)

par

Clément Forest

Directeur de thèse : Nicholas Ayache
Co-Directeur : Hervé Delingette

Composition du jury

Rapporteurs :	C. Chaillou
	Y. Payan
Examinateurs :	N. Ayache
	H. Delingette
	J-J. Levy
	L. Soler

Institut National de Recherche en Informatique et Automatique

Mis en page avec la classe thloria.

À Hélène,
à Mónica.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 La chirurgie endoscopique

Parmi les innombrables progrès qu'a connus récemment la chirurgie, le développement de la chirurgie minimalement invasive, ou chirurgie *endoscopique*, tient une place importante. L'endoscopie est une technique de diagnostic qui consiste à examiner l'intérieur des organes ou des cavités internes du corps, à l'aide d'un appareil appelé *endoscope*. L'intérêt de ces méthodes est que, comparées à des opérations ouvertes du même type, elles réduisent considérablement le traumatisme du patient. Cette amélioration est la conséquence directe de la forte diminution de la taille des incisions pratiquées, ou même de leur suppression, lorsque une voie d'accès naturelle existe. On observe ainsi d'une part une diminution des séquelles esthétiques liées à la taille des cicatrices, mais aussi d'autre part une chute spectaculaire de la durée totale d'hospitalisation. Les champs d'application de l'endoscopie sont très étendus : on les retrouve en orthopédie, en urologie, en gynécologie, en neurochirurgie, mais aussi en chirurgie vasculaire ou en chirurgie cardiaque. On emploie suivant les cas les termes d'arthroscopie, de bronchoscopie, de coloscopie, de gastroscopie, de cystoscopie ou encore d'hystéroscopie[1]. Dans le cas de la chirurgie digestive qui nous intéresse, on parle indifféremment de *cœlioscopie* ou de *laparoscopie*.

L'histoire de l'endoscopie moderne débute en 1806 avec l'invention par Philip Bozzini, chirurgien italien installé à Vienne, d'un appareil appelé *lichtleiter* (voir figure 1.2). Cet instrument comporte un système de miroirs qui permet l'envoi, à l'intérieur de la cavité d'un animal vivant, des rayons lumineux générés par une bougie, ainsi que leur retour jusqu'à l'œil du praticien. Réprimandé par la faculté de médecine pour sa "curiosité", Bozzini n'a pas pu tester son invention, véritable ancêtre de l'endoscope, sur un humain. En 1853, le français Antoine Jean Desormeaux améliore le liechtleiter de Bozzini et choisit d'utiliser une lampe à mèche comme source de lumière. Il augmente la visibilité mais cause, en contrepartie, des risques de brûlures. Desormeaux, qui utilisa son invention sur de nombreux patients, principalement dans le domaine de l'urologie, est généralement considéré comme le père de l'endoscopie. L'éclairage de la cavité est longtemps restée un problème crucial. En 1865, l'irlandais Cruise améliore l'intensité lumineuse en remplaçant l'essence et la térébenthine par un mélange de camphre et de pétrole. En 1873, le viennois Maximilien Nitze s'inspire de l'invention d'Edison et utilise un fil de platine chauffé à blanc par le passage d'un courant électrique, ce qui améliore encore l'illumination mais surtout diminue les risques de brûlures. C'est finalement l'arrivée de la lampe à incandescence à la fin du XIX^{ème} siècle, et surtout sa miniaturisation, qui permet l'apparition des premiers endoscopes à ampoule. Les progrès modernes et l'apparition des fibres optiques permettent de nos jours l'éclairage efficace et sans danger pour les cavités.

En 1869, Pantaleoni réalisa la première hystéroscopie, en 1879 Leistner pratique la première cystoscopie, en 1881 Miculicz la première gastroscopie et en 1918 Takagi la première arthroscopie du genou. Si les premières opérations se cantonnaient au

1. Il s'agit respectivement de l'examen des articulations, des poumons, du colon, de l'estomac, de la vessie, et de l'utérus.

diagnostic ou à la réalisation de biopsies, la tentation d'aller vers la thérapie et la chirurgie fut présente dès le début avec des actes de plus en plus complexes.

FIG. 1.1 – *Exemple de cours de chirurgie laparoscopique à l'IRCAD (sur des cochons). En haut à gauche, l'ensemble des outils utilisés. En bas à gauche on réalise le pneumopéritoine après avoir mis en place l'insufflateur. On peut observer au premier plan les deux trocarts qui vont accueillir les instruments et au fond celui qui sera utilisé par la caméra. L'image de droite montre la disposition du praticien et des aides lors de l'opération.*

La laparoscopie a été plus longue à franchir le pas et se cantonna longtemps dans le domaine du diagnostic. C'est en 1901 que George Kelling observe pour la première fois la cavité abdominale d'une chienne préalablement remplie d'air et c'est en 1911 que le suédois Hans Christian Jacobeus utilise le premier cette méthode sur un être humain pour observer le thorax et l'abdomen. Dans les années 50, le gynécologue Raoul Palmer découvre l'importance du maintien d'un pression constante dans la cavité abdominale. Le premier insufflateur automatique fut développé par l'allemand Kurst Semm en 1960. En 1982, la miniaturisation des caméras vidéos rend possible les premières opérations de *vidéo-laparoscopie* et ce n'est qu'en 1987 que le chirurgien lyonnais Philippe Mouret réalise la première *cholécystectomie* (ablation de la vésicule biliaire) par cœlio-chirurgie.

Les techniques de chirurgie laparoscopique sont en effet spécifiques et assez délicates à maîtriser. Par exemple, le chirurgien n'opère plus directement avec ses mains mais à l'aide de longs instruments dont l'axe passe par un point fixe, ce qui a pour conséquence d'inverser tous les mouvements. Le chirurgien ne regarde plus ses mains mais un écran et les informations de relief ne lui sont plus accessibles directement. Il

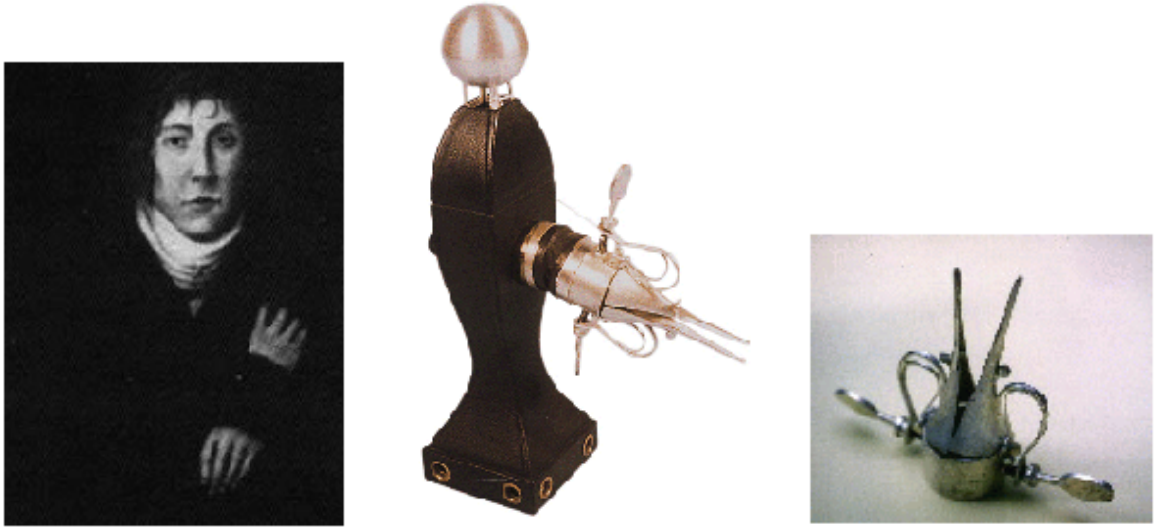


FIG. 1.2 – *Philip Bozzini (1773-1809) et le* Lichtleiter

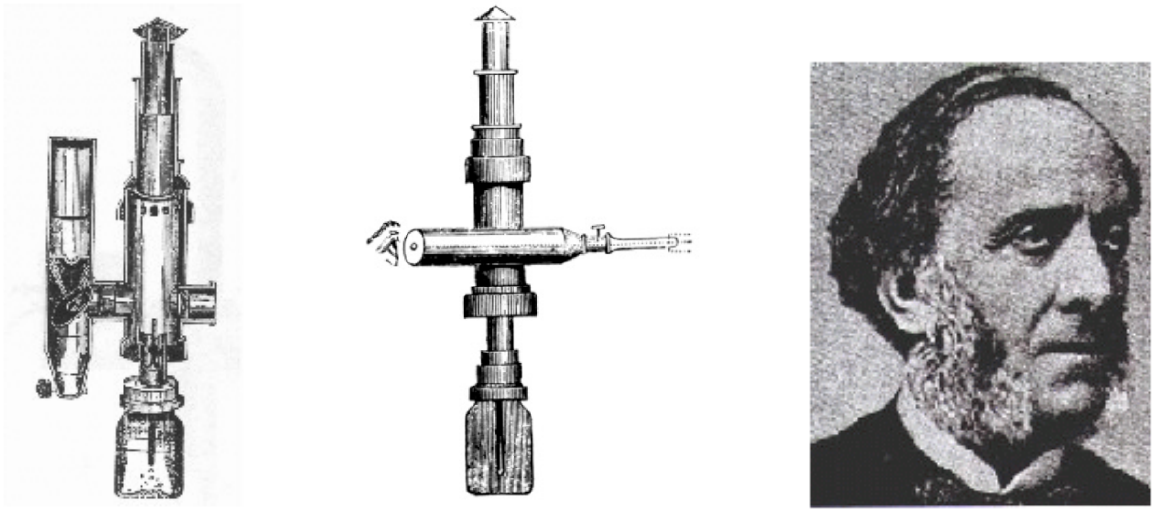


FIG. 1.3 – *L'endoscope de Desormeaux (1815-1882)*

lui est de plus difficile de s'orienter car la caméra est manipulée par un assistant à qui il doit demander d'effectuer les mouvement désirés.

1.2 Intérêt d'un simulateur de chirurgie laparoscopique

Actuellement, l'apprentissage des techniques de laparoscopie peut se faire de plusieurs manières différentes. Il est d'abord possible de s'entraîner sur un mannequin. Celui-ci peut être plus ou moins réaliste et permet de s'habituer au maniement des instruments et de pratiquer des gestes simples comme la suture. Son principal défaut est son manque de réalisme. Certains instituts, comme l'IRCAD à Strasbourg, proposent des sessions d'entraînement *in vivo* sur des animaux, généralement des cochons. Ces entraînements sont très efficaces car il s'agit d'une véritable opération, pratiquée dans des conditions réelles. Ils sont cependant peu nombreux car relativement chers. De plus l'utilisation d'animaux, qui doivent être sacrifiés en fin d'opération, pose de nombreux problèmes juridiques et est même interdite dans certains pays. Il existe aussi un certain nombre de différences anatomiques entre l'homme et le cochon qui limitent la portée de cet entraînement. Il est encore possible d'opérer sur des cadavres humains, mais on perd énormément de réalisme du fait de l'absence de saignement, de respiration et de battement cardiaque, et leur coût reste élevé. En fait, l'apprentissage est en majorité effectué directement sur le patient. L'interne assiste aux opérations et, au fil du temps, intervient de plus en plus, en restant sous le contrôle du praticien confirmé. Outre les risques pour le malade, cette méthode a le défaut de ne confronter l'étudiant chirurgien qu'aux pathologies que le hasard aura bien voulu mettre sur sa route.

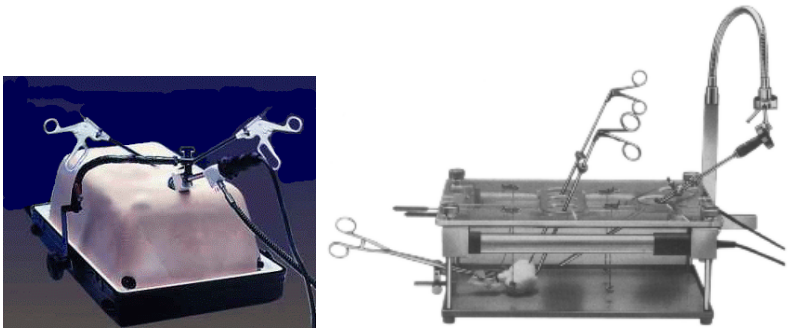


FIG. 1.4 – *Deux modèles de mannequins d'entraînement (Sources : gauche : Surgical Corporation, droite : ETZH, projet LASSO)*

De la même manière que les pilotes d'avion apprennent et se perfectionnent sur des simulateurs de vols, beaucoup prévoient dans un futur proche un usage courant des simulateurs de chirurgie [Marescaux *et al.*, 1998]. L'avantage de ces derniers sur les méthodes actuelles d'enseignement est que, pour un coût marginal très faible, ils

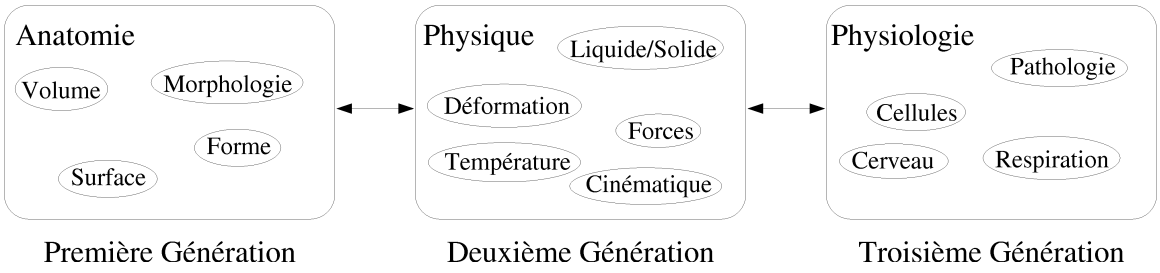


FIG. 1.5 – *Les trois générations de simulateurs proposées par Satava [Satava, 1996]*

permettent de faire répéter un grand nombre de fois les mêmes exercices par un ou plusieurs élèves différents, de les enregistrer pour plus tard les visionner, les commenter et les évaluer. On peut aussi imaginer disposer d'une bibliothèque de cas pathologiques constituant une base que tout étudiant se devrait de connaître et de maîtriser. De plus, et comme dans le cas des simulateurs aériens, il pourrait être possible au professeur de créer à volonté des évènements tels qu'une chute de la pression sanguine, une accélération du rythme cardiaque ou un réveil du patient, auxquels les étudiants devraient réagir correctement. Il existe encore d'autres avantages aux simulateurs. D'abord ils pourraient permettre à des praticiens confirmés de se remettre à niveau en pratiquant des actes qu'ils n'ont pas effectués depuis longtemps. S'ils deviennent suffisamment perfectionnés, ils pourraient même servir à préparer une opération réelle, en utilisant un modèle obtenu à partir de données provenant du patient à opérer. Enfin, et après un travail de validation poussé, on pourrait éventuellement imaginer s'en servir dans le cadre, de plus en plus problématique, des relations avec les assurances, pour justifier de la capacité de praticiens réputés confirmés.

1.3 Simulateurs existants

Le professeur R. Satava a proposé une classification des simulateurs en trois catégories [Satava, 1996]. Les simulateurs dits de *première génération* se contentent de décrire la géométrie des structures et des organes concernés. S'il est quelquefois possible d'interagir avec ceux-ci, l'interaction reste assez limitée. De très nombreux simulateurs de ce type sont présents sur le marché, en particulier des simulateurs de coloscopie, de trachéoscopie, d'arthroscopie ou encore d'échographie.

Les simulateurs de *seconde génération* considèrent de plus les comportements physiques tels que la mécanique, la température, la diffusion, etc. Il existe un grand nombre de simulateurs de ce type en cours de développement et certains commencent à être commercialisés. On peut noter la simulation de résection de la vésicule biliaire [Cover *et al.*, 1993], d'arthroscopie du genou [Gibson *et al.*, 1997], de chirurgie gynécologique [Székely *et al.*, 1999] ou encore d'introduction de cathéters. Ces simulateurs cherchent le plus possible à utiliser des valeurs réalistes pour les grandeurs caractéristiques. La conception d'appareils et d'expériences permettant d'obtenir ces valeurs reste cependant une tâche difficile.

FIG. 1.6 – *Le simulateur Mist de la société Mentice et le LapSim de la société Surgical Science (sources : www.mentice.com) et www.surgical-science.com)*

Les simulateurs de *troisième génération* prennent en compte des composantes physiologiques telles que la respiration, la pression sanguine, les battements cardiaques. On peut vouloir relier ces grandeurs entre elles et, par exemple, observer les conséquences d'une hémorragie sur la pression sanguine et sur la rigidité d'organes tels que le foie. Si aucun simulateur de ce type n'est actuellement commercialisé, plusieurs équipes travaillent ou ont travaillé sur le sujet : modélisation du système respiratoire [J. Kaye, 1998], du cœur, du cerveau, etc.

Le premier simulateur de chirurgie laparoscopique pourrait être le Minimally Invasive Surgery Trainer (*MIST*), développé dès 1995 par la société **Virtual Presence Ltd**. Cet outil se compose de deux pinces laparoscopiques, reliées à petit un ordinateur personnel, et ne disposant pas de capacité de retour d'effort. En fait, la cavité péritonéale n'est même pas représentée ce qui fait que le terme de simulateur ne peut être employé qu'entre guillemets. L'étudiant doit effectuer des exercices simples consistant à aller toucher une sphère bleue avec ses instruments et la placer à un autre endroit. La force et la popularité de MIST résident dans son environnement pédagogique. Les exercices sont conçus avec soin et cherchent à développer un certain nombre de compétences bien définies et à permettre l'évaluation de celles-ci.

Depuis cette date, un grand nombre d'entreprises se sont lancées dans la simulation de chirurgie. Des simulateurs d'arthroscopie permettant d'interagir avec un environnement rigide existant, par exemple, chez **Mentice**[2] ou **Boston Dynamics**[3]. Certains produits proposent des modèles déformables, généralement surfaciques. Le modèle de déformation le plus utilisé est le modèle masse-ressort, du fait de sa simplicité de mise en œuvre. Si elle est proposée, la découpe ne concerne que les organes surfaciques. L'opération la plus souvent simulée est la cholécystectomie, disponible sur le LapChole de la société **Xitact**[4], le LapSim de **Surgical Science**[5] ou encore le RLT de **ReachIn**[6]. On trouve aussi des simulateurs d'opérations gynécologiques ou

2. www.mentice.com
3. www.bdi.com/Virtual_Surgery.html
4. www.xitact.com
5. www.surgical-science.com
6. www.reachin.se

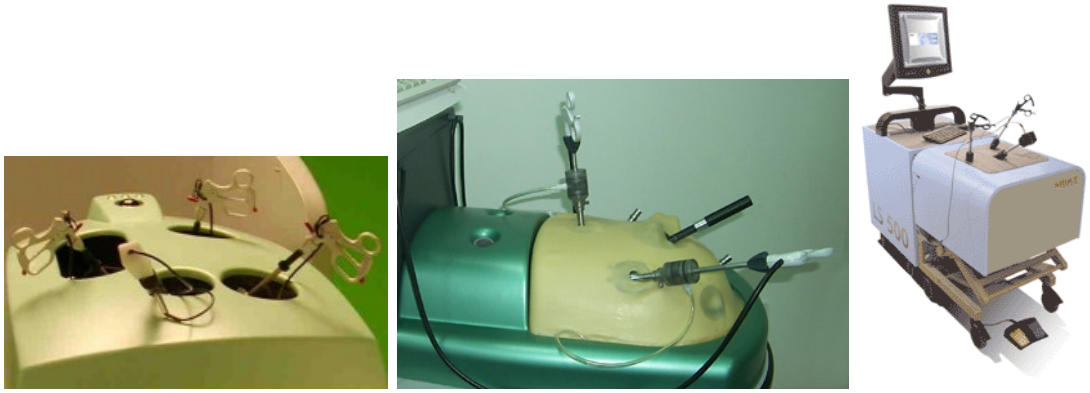


FIG. 1.7 – *Le VSOne de Select-It, le LapMentor de Simbionix et la plateforme LS500 de Xitact (source : www.select-it.de, www.simbionix.co.il et www.xitact.com)*

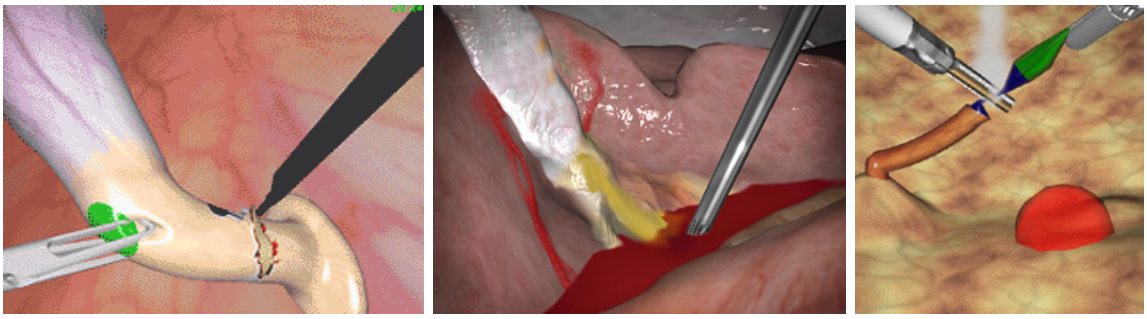


FIG. 1.8 – *Des vues du LapChole de Xitact, du LapMentor de Simbionix et du LapSim de Surgical Science (sources : www.xitact.com, www.simbionix.co.il et www.surgical-sciences.com)*

d'appendicectomie, comme par exemple avec le LapMentor de la société **Simbionix**[7] ou le VSOne de **SelectIt**[8]. Certaines de ces compagnies ont développé des plateformes matérielles utilisables aussi bien lors de la simulation d'opérations de chirurgie abdominale que gynécologique ou arthroscopique ; les différents scénarios sont alors disponibles sous forme de modules séparés. Le rendu visuel de plus en plus réaliste, du fait de l'utilisation de texture obtenues à partir d'images réelles, et les progrès dans les mécanismes de retour d'effort ont permis à ces produits d'acquérir une certaine maturité pouvant aller jusqu'à des tentatives de validation [Schijven and Jakimowicz, 2002].

La recherche académique s'est aussi beaucoup préoccupée de simulation de chirurgie et plusieurs projets ont d'ailleurs abouti à la conception de produits commerciaux. Les problématiques considérées sont diverses et arriver à toutes les intégrer au sein d'un même projet est déjà en soi une opération délicate. L'un des premiers centres d'intérêt est le développement de modèles mécaniques permettant le calcul des défor-

7. www.simbionix.co.il
8. www.select-it.de

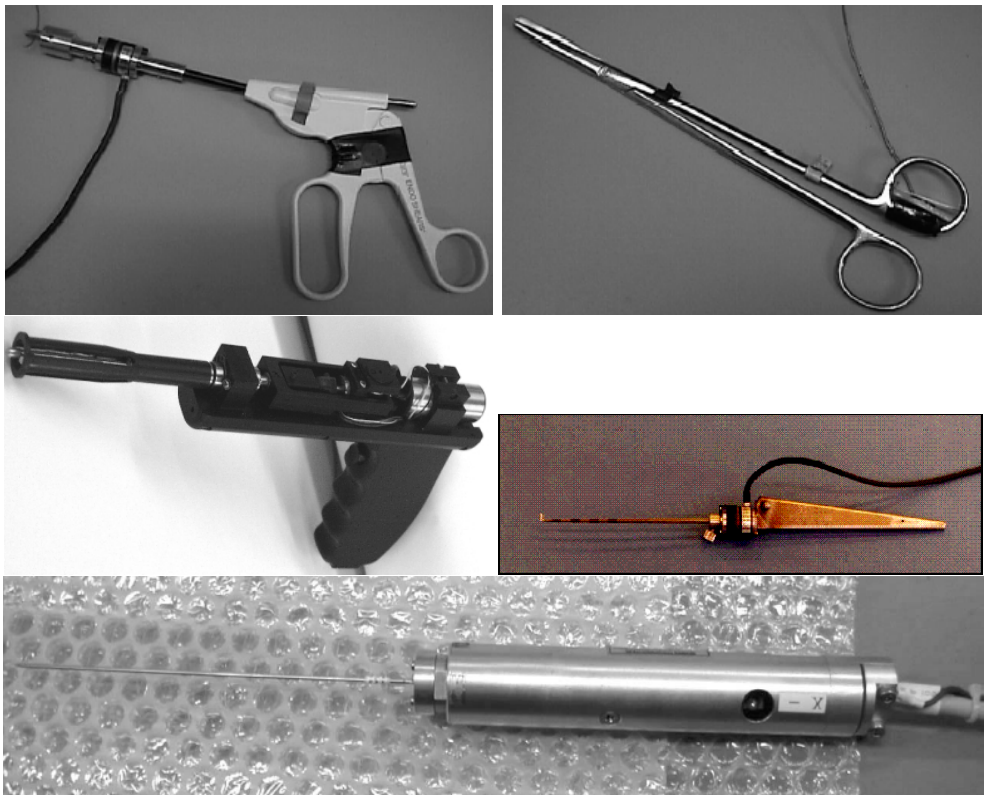


FIG. 1.9 – *Outils utilisés pour les mesures rhéologiques* in vivo *dans [Brouwer* et al.*, 2001] (haut), [Carter, 1998] (Milieu gauche) [Ward* et al.*, 1999] (Milieu droit), et [Kataoka* et al.*, 2002] (Bas).*

mations en temps réel. Certaines équipes utilisent des modèles masse-ressort [Brown *et al.*, 2001b; Bourguignon and Cani, 2000], d'autres leurs préfèrent des schémas plus complexes inspirés des travaux de Fung [Fung, 1993] ou de Maurel [Maurel *et al.*, 1991] et basés sur les éléments finis [Cotin, 1997; Picinbono, 2001] ou sur d'autres techniques : par exemple la méthode des éléments longs (*Long element Method*) [Costa and Balaniuk, 2001], celle des éléments radiaux (*Radial Element Method*) [Balaniuk and Salisbury, 2003] ou encore les méthodes utilisant des maillages adaptatifs [Wu *et al.*, 2001; Debunne, 2000; Debunne *et al.*, 2001]. Pour obtenir des déformations réalistes, la détermination de la valeur des constantes mécaniques est presque aussi importante que le modèle lui-même. Cette opération est délicate et peut nécessiter la conception d'outils particuliers. Plusieurs équipes ont tenté d'obtenir des telles données, que ce soit *in vivo*, de manière invasive [Brouwer *et al.*, 2001; Carter, 1998; Kataoka *et al.*, 2002; Bruyns and Ottensmeyer, 2002; Ottensmeyer and Salisbury, 2001; Kauer and Vuskovic, 1999] ou non invasive [Maaß and Kühnapfel, 1999; Ward *et al.*, 1999; Mazzella, 1999] ou encore *in vitro* [Dan, 1999; Schwartz *et al.*, 2002; Sakuma *et al.*, 2003]. L'interaction avec les outils virtuels est aussi l'objet de nombreuses discussions. Les techniques de détection de collisions sont fortement sollicitées pour la détermination des contacts entre outil et organe ou entre organes. Si les algorithmes utilisés sont souvent basés sur des méthodes classiques, telles que les sphères

imbriquées [O'Sullivan and Dingliana, 1999] ou les boîtes englobantes [van den Bergen, 1997], des techniques particulières ont été développées [Lombardo *et al.*, 1999; Aharon and Lenglet, 2002]. La détection de collision est particulièrement délicate lorsque l'on considère des objets pouvant s'auto intersecter, tels que les intestins [France *et al.*, 2002] ou les fils de chirurgie [Lenoir *et al.*, 2002]. La manière de traiter les collisions ainsi détectées est aussi étudiée [Kühnapfel *et al.*, 2001; Bruyns and Montgomery, 2002b]. Une grande attention est en particulier apportée à la découpe dynamique de maillage [Nakao *et al.*, 2002; Bruyns and Montgomery, 2002a; Nienhuys and van der Stappen, 2001] ou encore à son raffinement dynamique [Paloc *et al.*, 2002]. Certaines équipes se préoccupent aussi du développement de nouveaux systèmes à retour d'effort [Baumann *et al.*, 1997; Papadopoulos *et al.*, 2002] ou de l'amélioration de systèmes existants [Montgomery *et al.*, 2001]. On peut aussi citer des travaux sur les techniques de rendu graphique [Neyret *et al.*, 2002].

1.4 Simulateur Epidaure

1.4.1 Historique

Le projet de développer un simulateur de chirurgie à l'intérieur du projet Epidaure date de 1995 et coïncide avec le début de la thèse de Stéphane Cotin [Cotin, 1997], réalisée dans le cadre du projet européen MASTER (Minimal Access Surgery by Telecomunications and Robotics). Ces travaux ont débouché sur la conception d'un premier prototype du simulateur. Le modèle de déformation utilisé, dit *précalculé* [Cotin *et al.*, 1999], permet d'avoir une très grande fréquence d'itération, mais ne permet pas la découpe. Il propose aussi une méthode par éléments finis, appelée *masse-tenseur* [Cotin *et al.*, 2000], et basée sur un modèle d'élasticité linéaire. Cette méthode permet la découpe mais n'est applicable en temps réel que pour des maillages relativement simples. L'interaction s'effectue à l'aide d'un système à retour d'effort : le Laparoscopic Impulse Engine (*LIE*) de la société Immersion. L'intérêt de ces travaux est d'utiliser les techniques de reconstruction automatique déjà développées par le projet Epidaure pour obtenir des modèles d'organes réalistes [Cotin *et al.*, 1999]. Le projet s'est alors poursuivi dans le cadre d'une "action de recherche collaborative" appelée Action Incitative SIMulation de chirurgie (*AISIM* [9]) regroupant plusieurs projets de l'Inria (Epidaure, iMagis, Mostra, M3N, Sharp, Sinus), ainsi que de l'Institut de Recherche sur le Cancer de l'Appareil Digestif (*IRCAD*), intervenant en tant qu'expert médical. Cette action a permis de profiter des compétences de chacun et de considérer le problème sous différents aspects : le développement de modèles déformables, l'amélioration des méthodes de rendu graphique, l'interaction avec les modèles, l'utilisation des systèmes à retour d'effort, etc. Pour Epidaure, ce projet s'est concrétisé par la soutenance, en février 2001, de la thèse de Guillaume Picinbono [Picinbono, 2001] qui reprend la méthode des *masse-tenseur* et permet son utilisation en temps réel dans le cadre du simulateur. Il en propose aussi des ver-

9. `www-sop.inria.fr/epidaure/AISIM`

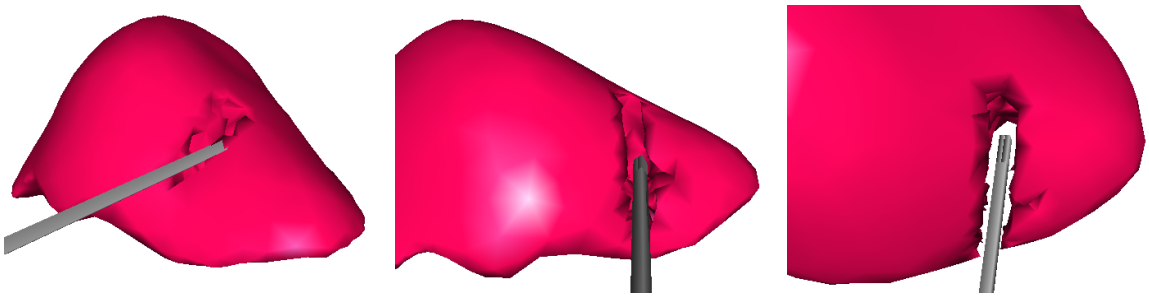


FIG. 1.10 – *Exemple de découpe du foie effectuée avec la version du simulateur de [Picinbono, 2001].*

sions anisotropes, ou non linéaires [Picinbono *et al.*, 2000; Picinbono *et al.*, 2001a; Picinbono *et al.*, 2001b; Picinbono *et al.*, 2002]. L'action AISIM s'est prolongée sous la forme d'une autre action appelée Chirurgie Abdominale Et Simulation À Retour d'Effort (*CAESARE*[10]), regroupant l'ensemble des partenaires d'AISIM avec, en plus, un partenaire industriel : l'ESI. L'idée de CAESARE était d'aboutir à un prototype utilisable dans le cadre de l'EITS, centre de formation rattaché à l'IRCAD.

1.4.2 Description de l'opération à simuler

L'opération que l'on désire simuler est la résection par cœlioscopie de l'un des segments anatomiques du foie. Les techniques que l'on emploie sont relativement génériques et peuvent s'adapter à d'autres formes de chirurgies endoscopiques ayant à manipuler des tissus mous. En particulier, des développements sont en cours pour appliquer ces résultats à la neurochirurgie.

Les techniques de laparoscopie sont très largement utilisées en chirurgie digestive : cholécystectomie bien sûr, mais aussi hernie hiatale ou splénectomie (ablation de la rate). Les exérèses hépatiques sont, elles, beaucoup plus rares. Leur manque de popularité est dû, d'une part aux risques spécifiques de ce genre d'opération, en particulier l'hémorragie et l'embolie gazeuse, et d'autre part aux difficultés techniques. Ces interventions consistent en la résection d'un ou de plusieurs des segments fonctionnels du foie : *les segments de Couinaud*. Ces segments, généralement au nombre de huit ou neuf, correspondent aux territoires vasculaires des embranchements principaux du réseau de la veine porte ; c'est pourquoi ils doivent être retirés en entier. Il est cependant possible de retirer certains kystes, situés en surface, sans procéder au retrait d'un segment entier. Les premières opérations de cœliochirurgie réalisées sur le foie concernaient justement ce genre d'opération, plus simples à réaliser. Pour le simulateur, nous ne nous intéresserons qu'aux opérations nécessitant une segmentectomie. Ce genre d'opération peut être rendu nécessaire lors de la présence de kystes, ou de tumeurs cancérigènes, situés profondément. Dans le cas des tumeurs, la nécessité de l'ablation de segments entiers est encore plus évidente, les métastases se propageant

10. `www-sop.inria.fr/epidaure/CAESARE`

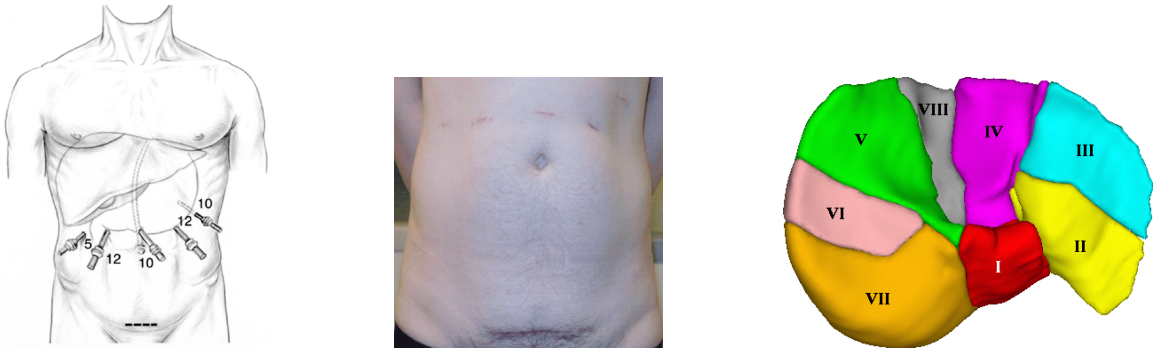


FIG. 1.11 – *Exemple de positionnement des instruments pour une résection des segments 2 et 3 du lobe gauche. Cicatrices résultant de cette même opération après un mois. Ce patient n'est resté que 5 jours à l'hôpital après l'opération et a pu reprendre le travail 3 semaines après (source [Cherqui* et al.*, 2002]). Exemple de segmentation du foie et d'étiquetage des segments de Couinaud (source [Soler, 1998])*

par le réseau sanguin. La résection de parties malignes n'est pas l'unique champ d'application de cette chirurgie : le professeur Cherqui, du CHU de Créteil, a par exemple réalisé très récemment deux opérations de prélèvement du lobe gauche par cœlioscopie dans des buts de transplantation intra-familiales [Cherqui *et al.*, 2002]. L'une des difficultés pratiques de la chirurgie hépatique par cœlioscopie est qu'elle ne permet qu'un accès partiel au foie. Il n'est pas possible de concevoir des opérations sur certaines zones hors de portée car placées trop en arrière : c'est typiquement le cas pour les segments numérotés I, VI et VII (voir figure 1.11). Pour améliorer l'accès à la zone de travail, il est souvent nécessaire de positionner le patient de manière adaptée, et éventuellement d'augmenter le nombre de trocarts utilisés. Les techniques d'imagerie permettent depuis quelque temps de parfaitement planifier et d'optimiser ces différents points avant le début de l'opération, en particulier depuis qu'ont été développées des techniques de segmentation automatique des segments de Couinaud [Soler, 1998].

La découpe du parenchyme peut s'effectuer à l'aide de différents instruments : le bistouri électrique, qui va coaguler automatiquement les parties ouvertes, le bistouri à ultrasons, qui va générer des vibrations à haute fréquence lesquelles vont pulvériser les cellules du parenchyme hépatique et laisser intactes les veines et autres structures moins denses en eau (voir figure 1.12). Les vaisseaux doivent en effet être traités avec soin car il faut limiter au maximum les pertes sanguines. De plus, du fait du pneumopéritoine au CO2, il existe des risques réels d'embolie gazeuse. Suivant leur taille, les vaisseaux sont donc simplement coupés et coagulés, ou encore préalablement clipés ou agrafés de part et d'autre avant d'être sectionnés. On utilise aussi d'autres instruments d'appoint tels que pinces, palpateurs, écarteurs, ou encore aspirateur pour manipuler le parenchyme et permettre l'évacuation du sang ou des déchets. L'extraction des segments découpés est aussi l'un des problèmes de cette méthode. Une cicatrice de trop grande taille rendrait en effet inintéressante l'usage de la cœlioscopie. Suivant les cas, on utilise une cicatrice déjà existante ou on effectue une incision sus-pubienne horizontale, moins dommageable esthétiquement et fonctionnellement

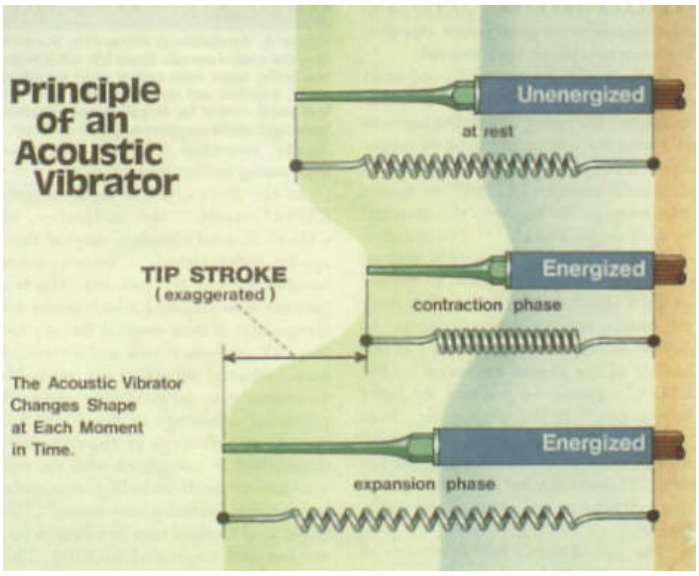


FIG. 1.12 – *Principe d'un bistouri à ultrason (source Universal Medical Press Inc.). La fréquence de vibration peut aller de 20 kHz à plus de 350 kHz suivant la nature des matériaux que l'on cherche à supprimer. Pour le parenchyme hépatique, les fréquences utilisées sont de l'ordre de 26 kHz. Le bistouri se compose de plus, d'une part d'un système d'irrigation, qui aide à désagréger le tissu et refroidit l'outil qui aurait tendance à chauffer sans cela, et d'autre part d'un système d'aspiration qui permet d'évacuer les déchets.*

qu'une ouverture classique. Il est aussi possible d'effectuer un carottage des parties à retirer ce qui permet d'évacuer la matière par l'un des trocarts.

1.4.3 Description du simulateur

Le simulateur développé au sein de l'équipe Epidaure se propose de simuler une résection hépatique et, plus particulièrement, de simuler la découpe du parenchyme, l'extraction des vaisseaux, leur traitement (pose de clips) et leur découpe. Les opérations situées en amont, telles que la planification, le placement des trocarts, la mise en place du pneumopéritoine, et celles situées en aval telles que l'extraction des segments résectés, ne font pas partie du simulateur, même si, pour certaines, elles pourraient éventuellement être intégrées à une version future.

Plusieurs modèles mécaniques peuvent être employés simultanément dans le simulateur. Pour les structures statiques, on peut employer des maillages surfaciques non déformables. Pour des objets qu'on peut manipuler, mais pas découper, on peut utiliser le modèle *précalculé* [Cotin, 1997; Cotin *et al.*, 1999]. Pour des organes que l'on désire découper, on peut utiliser l'une des mises en œuvre (linéaire ou non, isotrope ou non) du modèle *masse-tenseur* [Cotin, 1997; Cotin *et al.*, 2000; Picinbono *et al.*, 2000; Picinbono *et al.*, 2001a; Picinbono *et al.*, 2001b; Picinbono, 2001; Picinbono *et al.*, 2002] ou encore un simple modèle masse-ressort. Pour les structures comme les vais-

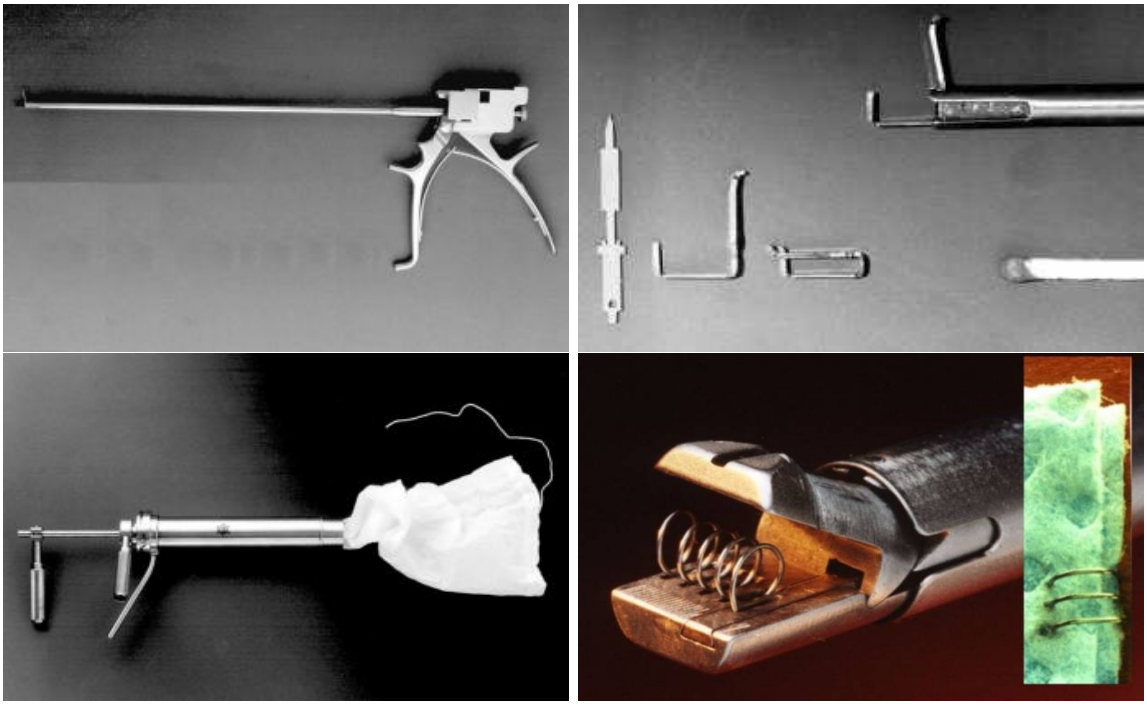


FIG. 1.13 – *Différents instruments utilisés lors d'une opération. Haut : outil permettant la mise en place d'agrafes pouvant servir à ligaturer des incisions. Bas gauche : extracteur pouvant servir à retirer les parties déjà réséquées du champ d'opération. Bas droite : pince permettant de suturer automatiquement des incisions. (source : www.fzk.de)*

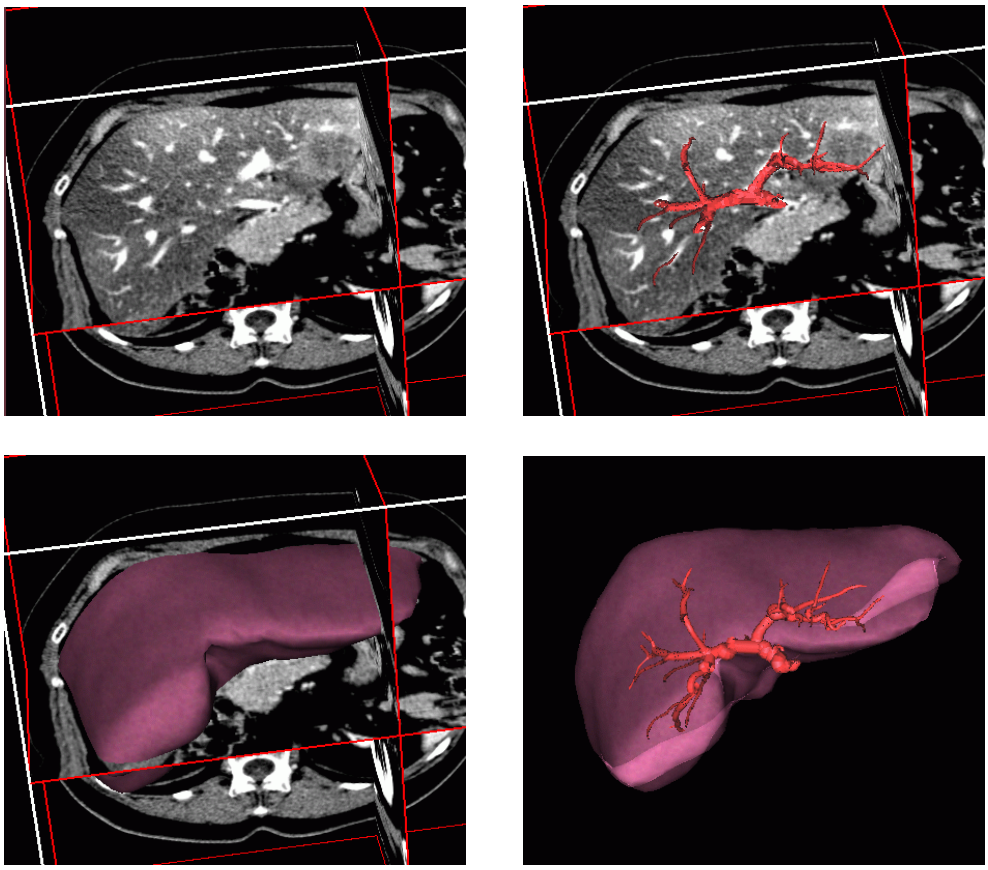


FIG. 1.14 – *Exemple de modèle géométrique (foie + veine porte) généré à partir du scanner d'un patient.*

seaux, on emploie un modèle linéaire (voir chapitre 5). Tous ces modèles peuvent interagir avec les systèmes à retour d'effort. Les modèles volumiques, surfaciques, ou linéaires, sont obtenus à partir de segmentations d'images tomodensitométriques ou d'IRM du patient, ce qui permet de disposer de modèles réalistes et variés, en particulier en ce qui concerne la position des tumeurs ou la répartition du réseau vasculaire à l'intérieur du parenchyme hépatique. Ces modèles sont représentés sous forme de maillages de façon à permettre l'application des schémas classiques de déformation. Pour les modèles linéaires, la forme choisie pour les cellules est le segment, pour les modèles surfaciques c'est le triangle et pour les modèles volumiques c'est le tétraèdre. Les raisons ayant conduit à l'utilisation d'éléments tétraédriques plutôt qu'hexaèdraux par exemple sont discutées au chapitre 2.3. On détaille ensuite la structure logicielle qui permet l'utilisation simultanée de plusieurs modèles de déformation. Pour des précisions sur ces modèles, en particulier sur les modèles *précalculés* et *masse-tenseur*, on pourra se reporter aux deux thèses précitées. On pourra aussi consulter le rapport de DEA de C.Checoury [Checoury, 2002], qui cherche à regrouper ces deux modèles à l'intérieur d'un même maillage, suivant une idée proposée par S. Cotin dans sa thèse [Cotin, 1997].

Le simulateur se base sur le logiciel `yav++` développé au sein de l'équipe Epidaure. Ce logiciel, écrit dans le langage `C++`, propose un grand nombre d'algorithmes formant une base de développement aux projets développés au sein de l'équipe. Il dispose d'une interface graphique et d'un interpréteur de scripts basé sur le langage TCL [Ousterhout, 1994]. Les principaux systèmes d'exploitation sont supportés par `yav++` : Linux, Windows, Irix, Digital Unix, SunOS. Le simulateur lui même fonctionne principalement sous Linux mais peut aussi être exécuté sous Irix ou sous Windows. Les différentes fonctionnalités du programme `yav++` sont disponibles sous la forme de modules indépendants qui peuvent être chargés dynamiquement en mémoire en fonction des besoins rencontrés.

Le simulateur doit partager ses ressources et le temps de calcul entre trois tâches différentes. La première est l'affichage graphique. Celle-ci doit s'effectuer à une fréquence minimum d'une trentaine de hertz et être suffisamment réaliste pour être crédible. Ce réalisme peut être amélioré par l'utilisation de textures, de lumière, ou encore par le multiplication du nombre de triangles affichés. La deuxième tâche est le calcul des déformations. Elle doit être effectué suffisamment rapidement de façon à obtenir un comportement réaliste. La fréquence minimum correspond évidemment à une seule itération entre chaque affichage de l'image. Dans le cas de modèles dynamiques, comme le masse-tenseur, une seule itération par pas de simulation peut conduire cependant à un comportement gélatineux et assez peu réaliste et il peut alors être souhaitable d'en effectuer plusieurs. La dernière tâche consiste en la gestion du retour d'effort, qui doit être exécutée sur un processus indépendant fonctionnant à une fréquence minimale de 500 Hz (voir chapitre 4).

Les versions précédentes, limitées par la puissance des machines, devaient être exécutées simultanément sur deux machines distinctes. La première s'occupait du calcul des déformations et du rendu graphique : originellement il s'agissait soit d'une sta-

FIG. 1.15 – *Vues du simulateur Epidaure (source Nice Matin pour la première)*

tion graphique SGI Onyx2, soit d'un ordinateur portable PC doté d'une très bonne capacité graphique. La seconde s'occupait de l'extrapolation des forces : il s'agissait d'un ordinateur PC de modèle courant. Les deux machines dialoguaient entre elles à travers un système de *socket TCP*. L'amélioration simultanée de la rapidité des ordinateurs (des processeurs à 2 GHz sont disponibles, d'autres 4 GHz sont annoncés), de la puissance des cartes graphiques (la carte accélératrice GeForce4 permet l'affichage de plus de 52 millions de triangles par seconde), et l'apparition de machines biprocesseur nous ont permis de faire fonctionner de manière satisfaisante le simulateur sur une seule machine. La version actuelle (à l'heure de la rédaction) se compose d'un ordinateur personnel PC biProcesseur utilisant des processeurs Pentium IV tournant à une fréquence de 1 GHz et fonctionnant avec le système d'exploitation Linux. Cet ordinateur a été équipé d'une carte graphique performante (GeForceIV Ti4600). Les systèmes à retour d'effort, au nombre de deux ou trois, sont reliés à la machine par des ports PCI. Cette configuration nous permet d'obtenir des fréquences de fonctionnement de l'ordre de 30 Hz pour des maillages de l'ordre d'un millier de tétraèdres et de 400 triangles à afficher.

1.5 Présentation du travail effectué

Le but de cette thèse était de perfectionner le simulateur existant dans l'équipe Epidaure de façon à rendre possible son utilisation par des élèves chirurgiens. En parallèle, il a fallu accompagner les migrations du code qui, après être passé du langage `C` au `C++`, a été refondu dans un nouvel environnement plus général au sein du programme `yav++`. Ce développement a été réalisé en parallèle sur plusieurs systèmes, en particulier Linux, Irix et Windows. L'une des plus grandes modifications qui ont été induites par le changement d'environnement a été la refonte profonde de la structure de données des maillages qui passait d'une représentation intuitive, basée sur de simple listes de primitives, à une représentation plus complexe, construite autour de la notion de *clôture* du maillage et de celle de *point virtuel*. Cette structure, présentée **au chapitre 2**, facilite énormément la construction et le parcours des voisinages, ce qui est une propriété très importante pour l'optimisation des algorithmes de calcul de déformation. Une autre propriété de cette structure de données est qu'elle permet

de mailler uniquement des variété, c'est-à-dire sans singularités topologiques. Si l'absence de singularités est apparue comme un point très intéressant par exemple pour le rendu graphique ou encore pour le calcul des forces à appliquer sur l'outil virtuel lors d'un contact avec le maillage, sa garantie impose de reconsidérer tous les algorithmes de modification du maillage. L'algorithme de découpe employé jusqu'à présent, et qui consistait simplement à retires du maillage les tétraèdres en contact avec la pointe active de l'outil virtuel, a en effet tendance à créer facilement des singularités et il a donc été nécessaire de modifier cet algorithme pour permettre d'une manière efficace le maintien du caractère variété du maillage. Un autre défaut de la découpe par retrait de tétraèdres est que, si la taille des tétraèdres retirés est trop grande, le résultat visuel est décevant. Pour résoudre ce problème, on propose une méthode de raffinement dynamique local. Le maillage est raffiné au voisinage de la pointe active de l'outil virtuel, préalablement à la découpe de façon à réduire la taille des tétraèdres retirés. Ce raffinement dynamique pose un certain nombre de problèmes. D'une part il faut limiter la dégradation de la qualité des nouveaux éléments afin d'éviter des problèmes de convergence lors du calcul des déformations. Cela est particulièrement vrai lorsque l'on raffine plusieurs fois des zones proches l'une de l'autre. D'autre part il faut transmettre à ces éléments des caractéristiques mécaniques cohérentes pour ne pas modifier le comportement global du maillage. Notre modèle de maillage servant de base à d'autres applications, on a mis au point des techniques génériques permettant ces transmissions. Les méthodes de modification des maillages (découpe et de raffinement dynamique) sont présentées **au chapitre 3**.

Ce simulateur doit permettre à l'utilisateur d'interagir avec les organes représentés. Il existe un certain nombre de périphériques permettant de simuler une pince de chirurgie et éventuellement de renvoyer la sensation d'effort à l'utilisateur. Si la détection des collisions entre outils et organes virtuels est une problématique toujours active, il est aussi important de savoir comment réagir lorsqu'une collision a été détectée. Ce point est encore plus délicat du fait que l'outil virtuel n'est pas ponctuel, comme c'est le cas classiquement, mais est assimilable à un segment passant par un point fixe. Les versions précédentes du simulateur faisaient l'hypothèse que la surface du maillage était régulière et relativement plane : le traitement des collisions et le retour d'effort devaient donc être désactivés dès que cette hypothèse n'était plus vérifiée. On propose une nouvelle méthode de résolution des collisions ne faisant pas d'autre hypothèse sur le maillage que celui de son caractère variété et pouvant donc être maintenue lors de la découpe. On a de plus développé une nouvelle méthode d'extrapolation, permettant l'envoi de force à 500 Hz, la simulation n'étant itérée qu'à environ 20 Hz. Cette méthode est en fait une application des méthodes dites de *modèle local* au cas d'un outil linéaire. Tous ces points sont développés **au chapitre 4**.

On a ensuite introduit les vaisseaux dans la modélisation du foie. Leur géométrie, obtenue après segmentation automatique d'une image scanner [Soler, 1998], conduit, après simplification, à un modèle unidimensionnel avec embranchements. Ce modèle dispose d'un habillage cylindrique, qui permet de percevoir son volume, mais qui ne complexifie aucunement les calculs de déformation. Ces calculs consistent simple-

ment en la résolution d'un système masse-ressort mais peuvent être complexifiés, par exemple par l'introduction de grandeurs telles que la torsion. Les arbres vasculaires sont alors introduits dans le maillage. Chacun des sommets du modèle unidimensionnel est plongé à l'intérieur de l'un des tétraèdres du maillage qui détermine sa position. Il lui transmet en retour une partie de sa force interne. Lors de la découpe, les tétraèdres contenant certains sommets peuvent être détruits. Ces sommets sont alors libérés et deviennent susceptibles d'être manipulés par l'outil virtuel (clampage, découpe). On présente les travaux relatifs aux vaisseaux **au chapitre 5**.

Une grande quantité de travail a aussi été nécessaire pour permettre la mise en œuvre réelle du simulateur. De nombreuses optimisations ou développements ponctuels se sont avérés indispensables. Par exemple, le modèle *masse-tenseur* [Cotin, 1997; Picinbono, 2001] que l'on utilise pour le calcul des déformations est très réaliste mais est relativement lent à converger. L'idée de la méthode que l'on propose est de résoudre le calcul de déformation non plus de manière globale, mais sommet par sommet. En jouant sur l'ordre ou la fréquence de résolution des sommets, on montre qu'il est alors possible d'accélérer sensiblement la vitesse de convergence du modèle. Dans un autre domaine, pour pouvoir utiliser sous le système d'exploitation Linux le périphérique à retour d'effort d'Immersion Corp que l'on utilise pour le simulateur, il a été nécessaire de développer un gestionnaire de périphérique dédié. On a aussi dû introduire un certain nombre de techniques d'accélération graphique afin d'améliorer l'aspect général du simulateur. Ces différents points sont développés **au chapitre 6**.

Au chapitre 7, nous ferons le bilan de nos travaux et nous évoquerons les projets concernant le simulateur de chirurgie. Finalement nous chercherons à dégager quelques unes des pistes qui devront être explorées dans le futur pour permettre aux simulateurs de chirurgie d'être aussi réalistes que peuvent l'être par exemple les simulateurs de vol.

1.6 Publications

Plusieurs articles, principalement centrés sur la problématique de la découpe, ont été publiés lors de cette thèse :

[**Forest *et al.*2002a**] Clément Forest, Hervé Delingette, and Nicholas Ayache. Cutting simulation of manifold volumetric meshes. In *Modelling & Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, 2002.

[**Forest *et al.*2002b**] Clément Forest, Hervé Delingette, and Nicholas Ayache. Cutting simulation of manifold volumetric meshes. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2488 of *LNCS*, pages 235–244, Tokyo, September 2002. Springer.

[**Forest *et al.*2002c**] Clément Forest, Hervé Delingette, and Nicholas Ayache. Removing tetrahedra from a manifold mesh. In *Computer Animation (CA'02)*, pages 225–229, Geneva, Switzerland, June 2002. IEEE Computer Society.

[**Forest *et al.*2003**] Clément Forest, Hervé Delingette, and Nicholas Ayache. Simulation of Surgical Cutting in a Manifold Mesh by Removing Tetrahedra. In *Medical Image Analysis*, soumis.

[**Sermesant *et al.*2002**] Maxime Sermesant, Clément Forest, Xavier Pennec, Hervé Delingette, and Nicholas Ayache. Biomechanical model construction from different modalities: Application to cardiac images. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2488 of *LNCS*, pages 714–721, Tokyo, September 2002. Springer.

1.7 Rappel des contributions

Au cours de cette thèse, nous avons :

- proposé une nouvelle structure de données pour les maillages volumiques permettant une détermination et un parcours efficace des voisinages,
- développé une technique de découpe de maillage volumique par retrait de matière utilisant un raffinement dynamique du maillage et garantissant la non apparition de singularités topologiques dans celui-ci,
- amélioré les interactions entre outil et maillage afin de pouvoir agir sur des surfaces irrégulières,
- adapté la méthode de retour d'effort dite du *modèle local* au cas d'un outil linéaire,
- introduit les vaisseaux dans le simulateur et permit leur manipulation,
- proposé des pistes pour une résolution asynchrone du calcul des déformations qui améliore la vitesse de convergence du modèle.

1.8 Conclusion

Même si plusieurs points restent encore à perfectionner, l'ensemble des travaux de cette thèse a permis de faire évoluer le simulateur jusqu'à rendre envisageable son utilisation dans un processus de validation clinique et d'industrialisation future. Une version de démonstration du simulateur fonctionne déjà depuis l'année dernière à l'IRCAD à Strasbourg.

Chapitre 2

Structure de données

Sommaire

2.1 Importance de la structure de données

Les bibliothèques de maillages volumiques et surfaciques du logiciel `yav++` ont été développées pour permettre la visualisation et la manipulation de modèles déformables volumiques et surfaciques. En particulier, elles proposent un mécanisme assez générique permettant de déformer ces maillages via l'application de forces et/ou de contraintes de position. Le choix d'un langage objet, tel que le `C++`, pour l'ensemble du projet `yav++`, permet en plus de profiter de la modularité induite par les notions de classes et d'héritage. Les maillages de base `Tetra3D` et `Triangulation3D` peuvent donc être spécialisés en fonction de l'utilisation désirée : `ActiveTetra3D` et `ActiveTriangulation3D` pour le caractère déformable, `PrecomputedTriangulation3D` pour les triangulations précalculées, `SimuTetra3D` pour la simulation de chirurgie, `HeartTetra3D` pour la modélisation du cœur, `BrainTetra3D` pour celle du cerveau, etc.

La détermination de la structure de données a une importance capitale. D'elle, dépend en effet la faisabilité et l'efficacité des opérations que l'on voudra réaliser. Il existe de nombreuses manières de juger une structure de données particulière. On en retiendra essentiellement trois :

1. **Efficacité pour l'exécution d'algorithme :** il s'agit du temps nécessaire pour réaliser les opérations désirées. Cela peut éventuellement inclure la création même de la structure de données. Pour notre modèle de maillage, l'une des opérations à optimiser est le parcours de voisinages, important lors des calculs de déformation.
2. **Encombrement mémoire :** il s'agit de l'espace mémoire nécessaire à la réalisation des opérations désirées. Cela inclut généralement l'espace mémoire nécessaire au stockage de la structure de données elle-même. Pour nous, cela nous incite à limiter au maximum la redondance dans les informations de voisinage.
3. **Facilité d'utilisation :** plus qualitative que les deux précédentes, elle correspond à la facilité qu'a le programmeur à utiliser la structure de données pour mettre en œuvre de nouvelles opérations. De trop grandes difficultés de mise en œuvre peuvent en effet fortement limiter les capacités d'évolution ou de maintenance d'une fonctionnalité donnée. À titre d'exemple l'introduction dans la structure de données de cartes permettant de retrouver efficacement l'arête ou le triangle reliant deux ou trois sommets donnés (cf. § 2.6.8) a facilité énormément l'écriture des opérations de modification du maillage (cf. § 2.7.2) et par là même a rendu possible la mise en œuvre de certaines d'entre elles, entre autres le raffinement local (cf. § 3.10.3).

Une structure de données n'est donc adaptée que dans le cadre d'une utilisation précise. La structure de données que l'on va présenter est efficace pour le calcul des déformations de maillages de taille moyenne (moins de 10.000 tétraèdres), mais n'est pas du tout performante si on cherche à s'en servir pour la visualisation de maillages très grands (par exemple de 100.000 triangles).

Avant de décrire avec précision la structure de données des maillages volumiques utilisés dans le simulateur de chirurgie, on propose un bref rappel sur les différentes manières de concevoir la modélisation d'objets volumiques sur ordinateur. La représentation sous forme de maillage volumique n'est en effet que l'une des différentes possibilités disponibles dans la littérature et de nombreuses mises en œuvre de simulateurs de chirurgie utilisent d'autres modèles.

2.2 Les différents modèles de représentation d'objets volumiques

Il existe un très grand nombre de modèles de représentation en machine d'un objet volumique donné. Historiquement les premiers objets volumiques modélisés sont le fait de la *Conception Assistée par Ordinateur* (C.A.O.), les ordinateurs étant en effet apparus comme un moyen très pratique de simplifier l'élaboration et l'édition des plans d'assemblage. Au fil des années est ensuite apparu le besoin de visualiser les objets ainsi modélisés, de les animer, d'en prévoir et d'en simuler le comportement, etc.

2.2.1 Modèles fil de fer

Cette méthode est probablement la première utilisée pour représenter un objet et date du début des années 50. L'objet est décrit comme une liste de points reliés par des arêtes, chaque arête pouvant être soit un segment de droite, soit une portion de courbe. Peu gourmande en espace mémoire et de représentation facile, cette approche permet en plus de calculer facilement des déformations à l'aide de modèles mécaniques de type *masse-ressort*. Son principal défaut est l'absence d'informations explicites sur la surface de l'objet ce qui peut conduire à des représentations ambiguës, c'est-à-dire pouvant correspondre à plusieurs volumes différents (cf. figure 2.1). De plus, pour des volumes de géométrie complexe, les représentations deviennent très vite visuellement peu compréhensibles.

2.2.2 Modèles constructifs

Les modèles constructifs ont été développés afin de faciliter les travaux de modélisation d'objets en CAO par exemple. La Géométrie Constructive des Solides (CSG, Constructive Solid Geometry) décrite par exemple par Mäntylä [Mäntylä, 1987] en est l'exemple le plus connu. Elle est encore disponible dans la quasi totalité des modeleurs volumiques, tel que PovRay. On la rencontre aussi en animation [Sanna and Montuschi, 1998] car les modèles obtenus sont facilement déformables, bien qu'il ne soit pas évident de leur appliquer des modèles de déformation réalistes.

Le volume est décrit comme la composition de formes de base simples (cylindres,

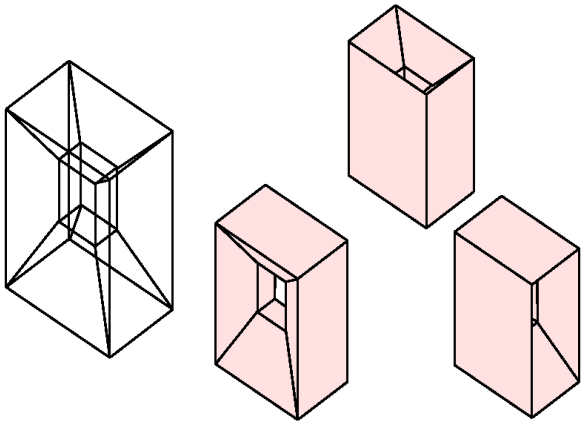


FIG. 2.1 – *Une représentation fil de fer ambiguë et les trois volumes auxquels elle peut correspondre*

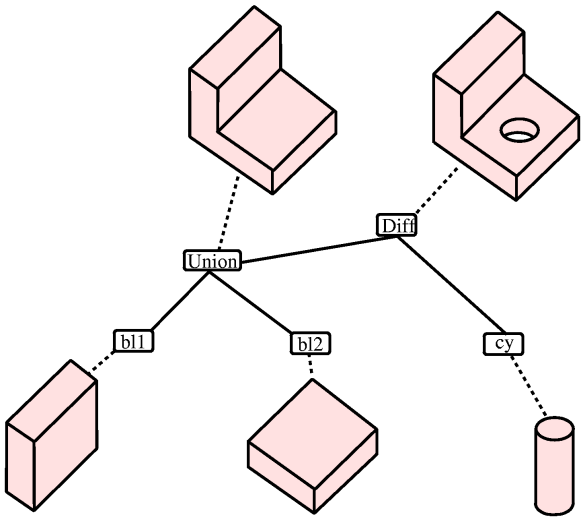


FIG. 2.2 – *Un arbre CSG*

cubes, tores, etc.) à l'aide d'opérateurs eulériens ($\bigcup$, $\bigcap$, $\setminus$, ...). Il est mémorisé comme un arbre dont les feuilles sont des primitives géométriques et les nœuds des opérateurs eulériens (cf. figure 2.2). Les représentations obtenues sont donc très compactes. La CSG est de plus extrêmement efficace pour la génération de nouvelles formes ainsi que pour certains calculs simples tels que l'appartenance d'un point au volume de l'objet décrit. Elle est cependant d'usage difficile pour la visualisation et celle-ci passe généralement par une transcription en une représentation par frontière (décrite ci-après).

2.2.3 Représentation par frontière

Les méthodes de Représentation par Frontière (B-Rep, Boundary Representation) sont nées du besoin de faciliter la visualisation d'objets préalablement modélisés à

l'aide de méthodes de type CSG. L'idée de ces méthodes est d'identifier l'objet à sa surface. Cette surface est décomposée en facettes, chaque facette pouvant être une portion de plan, de quadrique ou de quelque autre type de surface (tore, B-Spline, Bezier, NURBS [Krishnan and Manocha, 1996], ...), limitée par une ou plusieurs courbes fermées. On peut de plus diviser ces courbes en arêtes. Il existe plusieurs types de représentations par frontière, différant par les types d'arêtes ou de facettes dont on dispose. Ces méthodes permettent donc une description très précise des surfaces et peuvent être utilisées en mécanique (par exemple pour modéliser les ailes d'un avion). Elles peuvent être utilisée soit directement comme support à des modèles mécaniques volumiques, comme dans le cas des maillages précalculés [Cotin, 1997] ou de la méthode des éléments de surface (*Boundary Element Method*) [James and Pai, 1999], soit simplement comme enveloppe à de tels modèles [Debunne, 2000].

2.2.4 Subdivision spatiale

Énumération

Les méthodes d'énumération partitionnent l'espace en cubes à l'aide d'une grille régulière. Le solide est alors décrit comme l'union d'un certain nombre de ces cubes. Il existe plusieurs manières de représenter cette union. La première est l'énumération exhaustive, qui consiste pour chacun des cubes à dire si oui (*vrai*) ou non (*faux*) il appartient au solide. Une autre méthode, appelée *arbre octal* divise récursivement l'espace en huit cubes contigus de même taille, chacun de ces cubes n'étant divisé que s'il n'est pas uniformément vide ou plein. Ces méthodes sont intéressantes pour un très grand nombre de problématiques telles que l'intersection ou l'appartenance d'un point donné au solide. Elles sont cependant excessivement gourmandes en espace mémoire et ne permettent pas de mettre en œuvre simplement des modèles de déformations physiques. Les arbres octaux se rencontrent en visualisation [Srinivasan *et al.*, 1997], en détection de collision [Swan, 1993], ou encore lors de la génération de maillages [Perucchio *et al.*, 1989]. L'énumération exhaustive se rencontre naturellement dans les domaines du traitement d'images volumiques, par exemple médicales.

Décomposition en cellules

Cette approche partitionne le solide en éléments appelés *cellules*. Une cellule est une portion de volume homéomorphe à une sphère, c'est-à-dire sans trou. Deux cellules ne s'interpénètrent pas mais disposent d'un opérateur d'adjacence leur permettant de connaître leurs cellules voisines. Les maillages sont un cas particulier de cette décomposition ; ils seront décrits dans la partie 2.3.

2.3 Maillages volumiques

On ne parlera ici que de maillages volumiques. Les discussions suivantes peuvent cependant être transposées au cas des maillages de surfaces. Le cas échéant, on distinguera maillages volumiques et surfaciques en parlant respectivement de *tétraèdrisations* et de *triangulations*.

2.3.1 Définitions géométriques

On donne quelques définitions de géométrie sur les maillages, inspirées des ouvrages de P.J.Frey et P-L.George [Frey et George, 1999] ou de J.D. Boissonnat et M. Yvinec [Boissonnat et Yvinec, 1995]

Définition 2.3.1 *On appelle* polytope *l'enveloppe convexe d'un ensemble fini de points de $\mathbb{R}^3$. Soient A, B, C, et D quatre points indépendants, l'enveloppe convexe de A et de B est le segment de droite AB, celle des trois points A, B et C est le triangle ABC, celle des quatre points A, B, C et D est le tétraèdre ABCD.*

Définition 2.3.2 *Soit $\mathcal{P}$ un polytope. On appelle* hyperplan support *de $\mathcal{P}$ un hyperplan H tel que $H \cap \mathcal{P}$ soit non vide et tel que $\mathcal{P}$ soit inclus dans l'un des deux demi-espaces fermés définis par H.*

Définition 2.3.3 *L'intersection d'un polytope $\mathcal{P}$ avec l'un de ses hyperplans support est appelée* face *du polytope. Une face de dimension 2 est une* facette*, une face de dimension 1 est une* arête *et une face de dimension 0 est un* sommet *du polytope.*

Soit Ω un domaine borné de $\mathbb{R}^3$ et $\mathcal{T}$ une collection de polytopes de $\mathbb{R}^3$.

Définition 2.3.4 *La collection de polytopes $\mathcal{T}$ est un* maillage *du domaine Ω si :*
- *l'union de tous les polytopes de $\mathcal{T}$ est égale au domaine Ω,*
- *l'intérieur de tout élément K de $\mathcal{T}$ est non vide,*
- *l'intersection de l'intérieur de deux éléments de $\mathcal{T}$ est vide.*

Définition 2.3.5 *Le maillage $\mathcal{T}$ est un maillage* conforme *du domaine Ω si l'intersection de deux éléments de $\mathcal{T}$ est :*
- *soit l'ensemble vide,*
- *soit une face(facette, arête ou sommet) commune aux deux éléments en question.*

L'existence d'un maillage conforme est l'unique prérequis pour pouvoir appliquer la méthode des éléments finis. Les polytopes composant le maillage peuvent être de nature différente (tétraèdres, hexaèdres [Chabanas *et al.*, 2003; Székely *et al.*, 2000], dodécaèdre ou même prismes [Keeve *et al.*, 1996] etc.). Si le maillage combine des éléments de nature géométrique différente, il est dit *mixte*. Si on peut classer les

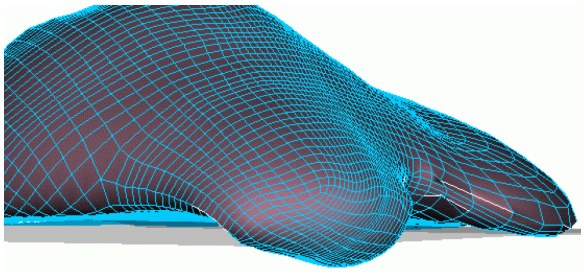


FIG. 2.3 – *Vue du maillage hexaédrique du foie maillé par la société ESI dans le cadre de l'action CAESARE*

éléments du maillage en lignes et colonnes, le maillage est dit *structuré*. Dans le cas du simulateur, le choix s'est porté sur des maillages tétraédriques non structurés. L'intérêt de l'utilisation des tétraèdres par rapport aux hexaèdres, d'usage parfois plus courant en mécanique, est multiple.

Premièrement, tout polytope peut être facilement décomposé en tétraèdres. Cette propriété permet aux mailleurs automatiques de générer très rapidement le maillage d'un domaine donné. À l'inverse il est très difficile de générer des maillages hexaédriques. Par exemple, il a fallu trois mois à la société ESI pour générer un maillage hexaédrique satisfaisant du foie (voir figure 2.3). Deuxièmement, même si les calculs sont moins précis qu'avec des éléments hexaédriques [Benzley *et al.*, 1995], ceux-ci sont grandement simplifiés [Picinbono, 2001, chapitres 2 et 3]. De plus, il est possible d'obtenir une expression analytique de la matrice de rigidité pour les maillages tétraédriques alors que pour les maillages hexaédriques on doit passer par une intégration de Gauss. Finalement, il est a peu près impossible d'opérer un remaillage local sur un maillage hexaédrique (sauf à perdre la conformité du maillage).

2.3.2 Définitions topologiques

Définition 2.3.6 *Soit $\mathcal{S}$ un ensemble de cardinal fini dont les éléments sont appelés* sommets. *Un quadruplet orienté (A,B,C,D), composé de quatre éléments différents de $\mathcal{S}$ est appelé* tétraèdre[11]. *Le tétraèdre (A,B,D,C) est appelé* tétraèdre inverse *du tétraèdre (A,B,C,D).*
Un triplet orienté composé de trois éléments différents de $\mathcal{S}$ est appelé triangle[12]. *Le triangle (A,C,B) est appelé* triangle inverse *du triangle (A,B,C).*
Une paire composée de deux éléments de $\mathcal{S}$ est appelé arête.

Définition 2.3.7 *Soit (A,B,C,D) un tétraèdre. Les trois triangles (A,B,C), (B,A,D), (C,D,A), et (D,C,B) sont appelés* triangles du tétraèdres. *Le triangle (D,C,B) est dit* opposé *au sommet A.*

11. Le quadruplet étant orienté, un tétraèdre possède 12 représentations différentes : (A,B,C,D), (A,C,D,B), (A,D,B,C), (B,A,D,C), (B,D,C,A), (B,C,A,D), (C,B,D,A), (C,D,A,B), (C,A,B,D), (D,C,B,A), (D,B,A,C) et (D,A,C,B)
12. Le triplet étant orienté, un triangle possède 3 représentations : (A,B,C), (B,C,A) et (C,A,B).

Définition 2.3.8 *Soient F un triangle et T un tétraèdre. On dit que F est* adjacent *à T si c'est l'un de ses triangles (*adjacence directe*) ou si son inverse est l'un de ses triangles (*adjacence inverse*) . Deux tétraèdres sont dits* adjacents *s'il existe un triangle adjacent aux deux tétraèdres.*

Soit $\mathcal{T}$ un ensemble de tétraèdres. On notera respectivement $\mathcal{S}_\mathcal{T}$, $\mathcal{E}_\mathcal{T}$ et $\mathcal{F}_\mathcal{T}$ l'ensemble des sommets, arêtes et triangles des tétraèdres de $\mathcal{T}$.

Définition 2.3.9 *L'ensemble des tétraèdres de $\mathcal{T}$ contenant un sommet donné A (resp. une arête donnée E) est appelé* voisinage du sommet A dans $\mathcal{T}$ *(resp.* voisinage de l'arête E dans $\mathcal{T}$*) et est noté $\mathcal{V}_A$ (resp. $\mathcal{V}_E$). Ce voisinage est dit* complet *si tous les triangles des tétraèdres de ce voisinage, concourants à A (resp. adjacents à E), sont adjacents à exactement deux tétraèdres. Ce voisinage est dit* connexe par les triangles *si on peut relier chaque paire de tétraèdres de ce voisinage par une chaîne de tétraèdres appartenant à ce voisinage et adjacents deux à deux.*

Observation 2.3.10 *On étend la définition précédente aux cas des sommets, des arêtes et des triangles. On dira que l'arête E ou que le triangle F appartiennent au voisinage du sommet A s'ils sont concourants à ce sommet. On dira que le sommet B appartient au voisinage du sommet A si l'arête (A,B) appartient à $\mathcal{S}_\mathcal{T}$.*

Définition 2.3.11 *L'ensemble $\mathcal{T}$ est appelé* maillage topologique *si :*

- *tous les triangles des tétraèdres $\mathcal{T}$ sont adjacents au plus à deux tétraèdres de $\mathcal{T}$, l'un de manière directe, l'autre de manière inverse,*
- *les voisinages des arêtes et des sommets des tétraèdres de $\mathcal{T}$ sont connexes par les triangles.*

Les sommets, arêtes et triangles des tétraèdres de $\mathcal{T}$ sont appelés sommets, arêtes et triangles du maillage*. Le maillage est dit* complet *si chacun de ses triangles est adjacent à exactement deux tétraèdres.*

Observation 2.3.12 *Si le maillage est complet, alors les voisinages de tous ses sommets et de toutes ses arêtes le sont aussi.*

2.3.3 Modèle de maillage utilisé dans le simulateur

On présente maintenant le modèle de maillage utilisé dans la bibliothèque `libTetrahedrisation`. On donne à cette structure le nom de *maillage topologiquement complet* pour la différencier du *maillage géométrique* de la définition 2.3.4 ainsi que de celle du *maillage topologique* de la définition 2.3.11.

Définition 2.3.13 *Soit Ω un domaine borné de $\mathbb{R}^3$. Soit $\mathcal{T}$ un ensemble de tétraèdres de $\mathbb{R}^3$. Soit $\mathcal{S}$ l'ensemble des sommets de $\mathcal{T}$. On dit que $\mathcal{T}$ est un* maillage topologiquement clos *du domaine Ω si :*
(H1) l'ensemble $\mathcal{T}$ est un maillage topologique fermé, au sens de la définition 2.3.11,

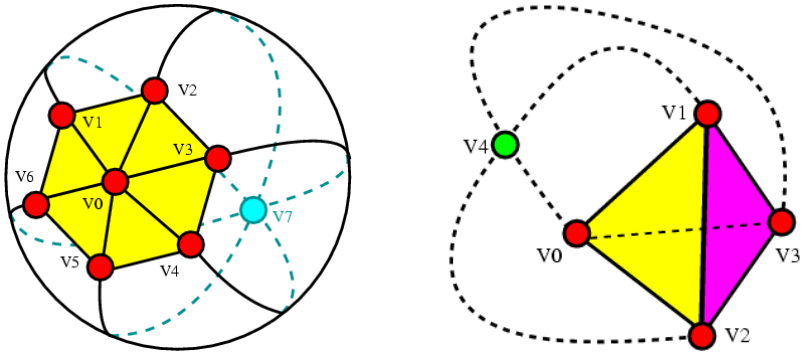


FIG. 2.4 – *Fermetures d'un maillage surfacique et d'un maillage volumique à l'aide de sommets virtuels*

(H2) il existe un sous-ensemble $\mathcal{T}_r$ *de* $\mathcal{T}$ *tel que* $\mathcal{T}_r$ *soit un maillage conforme du domaine* Ω*, au sens de la définition 2.3.5. Les éléments de* $\mathcal{T}_r$ *sont appelés* tétraèdres réels*, ceux de* $\mathcal{T}_v = \mathcal{T} \backslash \mathcal{T}_r$ *sont appelés* tétraèdres virtuels*,*

(H3) l'ensemble $\mathcal{T}_v$ *est exactement le voisinage topologique d'un sous-ensemble* $\mathcal{S}_v$ *de* $\mathcal{S}$*. Les éléments de* $\mathcal{S}_v$ *sont appelés* sommets virtuels*, ceux de* $\mathcal{S}_r = \mathcal{S} \backslash \mathcal{S}_v$ *sont appelés* sommets réels*,*

(H4) chaque tétraèdre virtuel est adjacent à exactement un tétraèdre réel,

(H5) un sommet de $\mathcal{S}_r$ *n'est relié par un tétraèdre au plus qu'à un seul sommet virtuel.*

Pour simplifier, la définition précédente revient à compléter le maillage géométrique en lui ajoutant un certain nombre de *tétraèdres virtuels* reliant les triangles de la surface à un sommet appelé *sommet virtuel*.

Observation 2.3.14 *Deux sommets virtuels ne sont jamais reliés entre eux par un tétraèdre*

Preuve : en effet, tous les sommets d'un tétraèdre réel sont réels (H3). Les tétraèdres virtuels ayant exactement un voisin réel (H4), ils possèdent chacun trois sommets réels. Il n'existe donc pas de tétraèdres reliant deux sommets virtuels. ⋄

Observation 2.3.15 *La surface du maillage est composée de l'ensemble des triangles opposés aux sommets virtuels.*

Preuve : ce sont en effet exactement les triangles adjacents à un unique polytope dans le maillage géométrique $\mathcal{T}_r$. ⋄

Observation 2.3.16 *Il y a exactement un sommet virtuel par composante connexe de la surface.*

Preuve : les triangles réels de deux tétraèdres virtuels adjacents partagent deux sommets et sont donc deux triangles adjacents de la surface du maillage. Le voisinage de chaque sommet virtuel étant complet, l'ensemble des triangles opposés à un sommet virtuel donné est une surface fermée. Ce voisinage étant de plus connexe par les triangles, cette surface est connexe par les arêtes. Il s'agit donc exactement de l'une des composantes connexes de la surface du maillage. ⋄

2.4 Notion de maillages variété

La notion de *maillage variété* a été principalement étudiée dans le cadre des maillages surfaciques [Guéziec *et al.*, 2001]. Elle permet en effet l'application d'un grand nombre d'algorithmes, principalement graphiques : rendu Gouraud, textures de lumières [Woo *et al.*, 1997, chapitre 9], PN triangles [Vlachos *et al.*, 2001], surfaces de subdivisions [Qin, 2000], etc. ; on retrouve cette notion en conception de matériaux [Bøhn, 1995] en compression [Taubin *et al.*, 1998] ou en raffinement de maillages [Brown, 1998]. La variété d'un maillage volumique est évoquée dans certains ouvrages à visée plus mathématique [Henle, 1979], ou cherchant à généraliser les définitions à la dimension n [Boissonnat et Yvinec, 1995; Hoffmann, 1989], mais cette notion possède peu d'applications.

Les définitions suivantes sont principalement inspirées du livre de Christoph M. Hoffmann [Hoffmann, 1989].

Définition 2.4.1 *Soit d et n deux entiers vérifiant $n \leq d$. Un sous-espace M de $\mathbb{R}^d$ est une* n-variété topologique *si tout point de ce sous-espace possède un voisinage homéomorphe à $\mathbb{R}^n$. Un sous-espace M de $\mathbb{R}^d$ est une* n-variété topologique avec bord *si tout point de ce sous-espace possède un voisinage homéomorphe, soit à $\mathbb{R}^n$, soit au demi-espace $\mathbb{R}^{n+} = \{(x_1,...,x_n) \in \mathbb{R}^n | x_1 \geq 0\}$*

Observation 2.4.2 *Pour un sous-espace variété avec bord, on peut faire la différence entre* points intérieurs*, pour lesquels le voisinage est homéomorphe à $\mathbb{R}^n$, et* points de la frontière*, pour lesquels le voisinage est homéomorphe à $\mathbb{R}^n+$. Pour les variétés sans bord il n'y a que des points intérieurs. D'autre part, la frontière d'une n-variété avec bord est une (n-1)-variété sans bord.*

On va maintenant étendre la notion de variété aux maillages.

Définition 2.4.3 *Un maillage $\mathcal{T}$ est une* variété *(avec ou sans bord) si le domaine Ω défini par l'union de ses cellules est une variété (avec ou sans bord).*

Observation 2.4.4 *Un maillage surfacique conforme est une variété sans bord si et seulement si :*

– *chaque arête est adjacente à exactement deux facettes,*

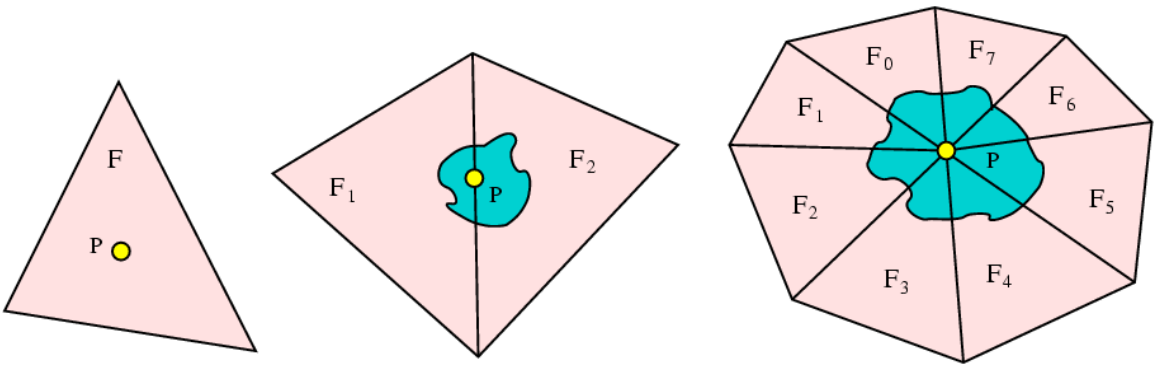


FIG. 2.5 – *Les trois cas possibles pour le point P*

– *le voisinage de chaque sommet est connexe par les triangles.*

Preuve : Soit P l'un des points de cette surface Ω. Montrons qu'il est possible de construire un voisinage de P homéomorphe au disque ouvert $\mathbb{S}_2$ (voir figure 2.5).

- Supposons que P appartienne à l'intérieur d'une facette F de $\mathcal{T}$. Soit $\mathcal{V}_F$ un voisinage ouvert du point P dans la facette F. Ce voisinage $\mathcal{V}_F$ est évidemment homéomorphe à $\mathbb{S}_2$. D'autre part, le maillage étant conforme, l'intérieur de la facette, et donc $\mathcal{V}_F$, n'est intersecté par aucune autre facette du maillage. $\mathcal{V}_F$ est donc aussi un voisinage de P dans Ω.
- Supposons que P appartienne à l'intérieur d'une arête de $\mathcal{T}$ (*ie.* ne soit pas l'un de ses sommets). Soient F_1 et F_2 les deux facettes de $\mathcal{T}$ adjacentes à cette arête. Soient $\mathcal{V}_{F_1}$ et $\mathcal{V}_{F_2}$ des voisinages ouverts de P dans chacune de ces deux facettes. L'intérieur de l'union de ces deux voisinages est un voisinage de P dans $F_1 \cup F_2$ homéomorphe à $\mathbb{S}_2$. Le maillage étant conforme, cette union n'est intersectée par aucune autre facette de $\mathcal{T}$ et constitue donc bien un voisinage de P dans Ω.
- Supposons pour finir que P soit l'un des sommets du maillage. Soit F_0 l'une des facettes de $\mathcal{T}$ concourantes à P. Par hypothèse, les arêtes de $\mathcal{T}$ étant adjacentes exactement à deux facettes, il existe exactement deux facettes adjacentes à la fois à F_0 et à P. Soit F_1 l'une de ces deux facettes. De proche en proche, on peut ainsi construire un ensemble $(F_i)_{i\in[0,n]}$ de facettes concourantes à P et adjacentes deux à deux (F_i avec F_{i+1} et F_n avec F_0). Soient $(\mathcal{V}_{F_i})_{i\in[0,n]}$ des voisinages de P dans chacune de ces facettes. L'intérieur de l'union de ces voisinages est un voisinage de P dans $\bigcup_{i\in[0,n]} F_i$ homéomorphe à $\mathbb{S}_2$. Le voisinage de P étant supposé connexe par arêtes, les F_i représentent la totalité des facettes de $\mathcal{T}$ adjacentes à P. De plus, le maillage étant conforme, aucune autre facette n'intersecte les F_i et cette union constitue donc bien un voisinage de P dans Ω.

Réciproquement, si $\mathcal{T}$ est un maillage conforme ne vérifiant pas l'une des deux hypothèse précédentes, il est facile de se persuader que le domaine Ω présente une singularité située sur l'arête ou sur le sommet correspondant et qu'il n'est donc pas une variété (voir figure 2.6). $\diamond$

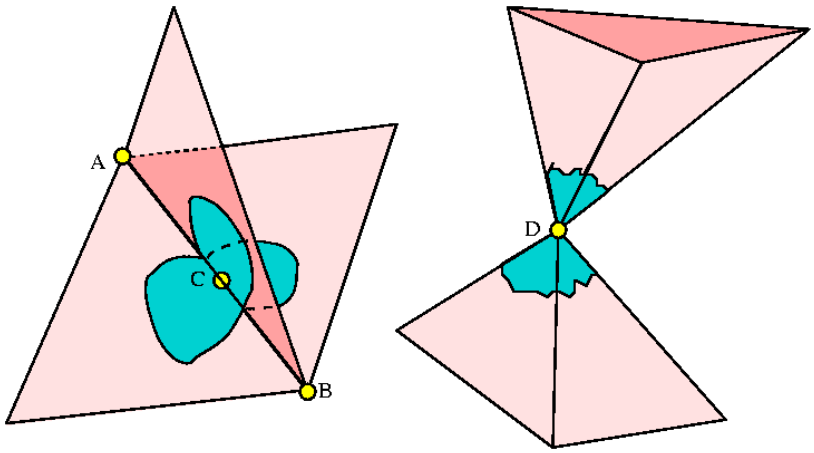


FIG. 2.6 – *L'arête AB est adjacente à trois facettes, les voisinages géométriques du sommets C ne peuvent donc pas être homéomorphe au disque ouvert de* $\mathbb{S}_2$. *Le voisinage du sommet D n'étant pas connexe par les triangles, ses voisinages géométriques ne peuvent donc pas être homéomorphe au disque ouvert de* $\mathbb{S}_2$.

Observation 2.4.5 *Un maillage volumique conforme est* une variété avec bord *si sa surface est un maillage variété.*

Preuve : En effet, les sommets de l'intérieur du maillage sont situés dans l'intérieur du domaine de celui-ci. Ils posèdent donc bien un voisinage homéomorphe à $\mathbb{R}^3$ et ne présentent donc pas de singularités. ◇

Observation 2.4.6 *La partie réelle d'un maillage topologiquement complet est forcement une variété.*

Preuve : Soit $\mathcal{T}$ un maillage topologiquement clos. Il suffit de montrer que la surface de $\mathcal{T}$ vérifie les trois hypothèses de 2.4.4.

- L'hypothèse H1 de la définition 2.3.13 impose le caractère conforme de maillage sous-jacent à $\mathcal{T}$ et donc de la surface de celui-ci.
- Soient E une arête de la surface de $\mathcal{T}$et A et B les sommets de cette arête. Soient $(F_i)_{i\in[1,n]}$ les facettes de la surface de $\mathcal{T}$ adjacentes à E, n étant le nombre de ces facettes. Montrons que $n = 2$. Par définition, chacune des facettes F_i est adjacente à un tétraèdre virtuel et ces tétraèdres sont tous différents (H4). Soit $(T_i)_{i\in[1,n]}$ l'ensemble de ces tétraèdres. Du fait de l'hypothèse H5, le sommet v_1 ne peut être relié au plus qu'à un seul sommet virtuel. Tous les tétraèdres T_i sont donc adjacents au triangle (A,B,V) où V est l'unique sommet virtuel relié à E. Le maillage $\mathcal{T}$ étant un maillage topologique (H1), ce triangle ne peut être adjacent qu'à deux tétraèdres et on a donc $n = 2$.
- Soit A un sommet de la surface de $\mathcal{T}$ et E_v l'arête virtuelle à laquelle il est relié. À chaque tétraèdre virtuel adjacent à E_v correspond un triangle de la surface adjacent à v. De même, à deux tétraèdres virtuels adjacents correspondent deux

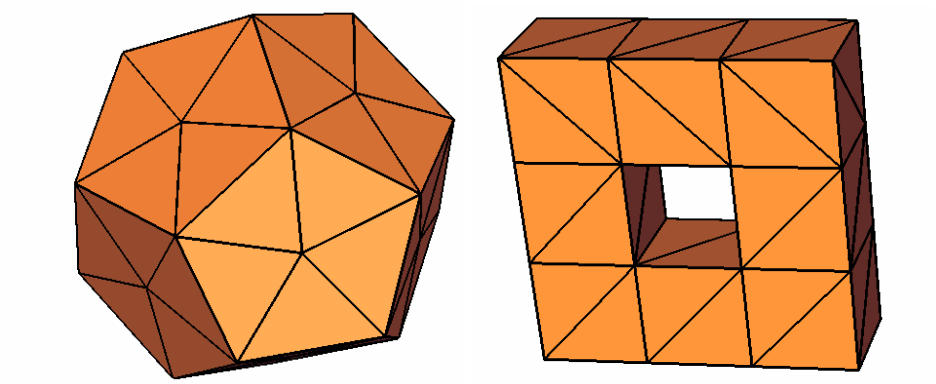


FIG. 2.7 – *Le maillage figure de gauche contient 34 sommets (dont 1 virtuel), 165 arêtes (dont 32 virtuelles), 262 triangles (dont 90 virtuels) et 131 tétraèdres (dont 60 virtuels). Sa caractéristique d'Euler vaut* $34 - 165 + 262 - 131 = 0$. *Celui de droite contient 33 sommets (dont 1 virtuel), 144 arêtes (dont 32 virtuelles), 224 triangles (dont 96 virtuels) et 112 tétraèdres (dont 64 virtuels). Sa caractéristique d'Euler vaut* $33 - 144 + 224 - 112 = 1$. *Il n'est donc pas homéomorphe à une 3-variété de* $\mathbb{R}^4$.

triangles adjacents. $\mathcal{T}$ étant un maillage topologique, le voisinage de E_v est connexe et c'est donc aussi le cas du voisinage de A.

◇

Observation 2.4.7 *Si un maillage surfacique est une variété, alors on peut définir une normale à la surface pour tous les sommets de ce maillage.*

Preuve : Cela est dû au fait que le voisinage des sommets sur la surface du maillage est simplement connexe. ◇

Observation 2.4.8 *Une triangulation topologiquement complète est homéomorphe à une surface variété de* $\mathbb{R}^3$.

Preuve : Il suffit de le plonger dans $\mathbb{R}^3$ et de rendre réel ses sommets virtuel. On se retrouve alors exactement dans le cadre de la définition 2.4.3. ◇

Observation 2.4.9 *Un tétraèdrisation topologiquement complète n'est pas forcément homéomorphe à une 3-variété de* $\mathbb{R}^4$.

Preuve : Si cela est bien le cas pour les solides de genre 0 (ie. sans trou), dans le cas d'un tore par exemple, on observe une singularité située sur le sommet virtuel. On peut aussi se persuader en remarquant expérimentalement que la tétraèdrisation topologiquement complète d'un tore n'a pas une *caractéristique d'Euler* [13] nulle. ◇

13. La caractéristique d'Euler $\chi = V - E + F - S$, où V, E, F, S sont respectivement le nombre de sommets, d'arêtes, de facettes et de cellules d'un maillage donné ne dépend pas de la manière dont le domaine est maillé et est nulle pour les variétés de dimension 3 [Henle, 1979].

2.5 Structures de données de maillages

2.5.1 Généralités

Un maillage comporte des informations de plusieurs types. D'abord des informations de connexité, c'est-à-dire sur sa topologie. Ensuite des informations géométriques : il s'agit essentiellement de la position des sommets mais cela peut aussi comporter d'autres informations telles que la valeur des normales à la surface, la longueur des arêtes, etc. Enfin il peut aussi contenir des informations spécifiques à l'utilisation que l'on cherche à en faire : coordonnées de texture, constantes de raideur, vecteurs d'anisotropie, etc. Les discussions sur les structures de données de maillage se focalisent principalement sur la manière de mémoriser la composante liée à la connexité, les autres informations n'étant généralement pas compressibles. De plus, elles ne concernent généralement que les maillages surfaciques et il existe relativement peu de littérature sur le sujet. On peut distinguer trois grandes classes de structures de données pour les maillages [Danovaro, 2001] :

- les structures de données qui se contentent de mémoriser la géométrie, c'est-à-dire la position des sommets de chaque cellule, mais ne fournissent aucune information d'adjacence. Les sommets, arêtes (et éventuellement triangles pour les maillages volumiques) ne sont pas explicitement créés et les informations correspondantes sont stockées directement dans les triangles (ou les tétraèdres). Pour ces structures, les données de connectivité ne sont pas mémorisées mais peuvent être retrouvées en comparant la position des sommets,
- les structures de données qui fournissent uniquement les relations de connexité entre les cellules et leurs sommets. Les sommets sont explicitement créés. La détermination des voisinages est réalisable en temps constant (proportionnel à la dimension du maillage),
- les structures de données qui fournissent explicitement des informations de voisinage. Pour un maillage tétraédrique, les tétraèdres possèdent donc des pointeurs vers chacun de leurs quatre tétraèdres voisins [Yao and Taylor, 2000] et, lorsqu'ils existent, vers chacun de leurs triangles et de leurs arêtes. Quelques fois les tétraèdres ne possèdent pas de pointeurs vers leurs voisins et on doit alors aller chercher l'information dans le triangle correspondant [Nienhuys and van der Stappen, 2001]. Les parcours dans le maillage se font en temps constant et la détermination des voisinages est facilitée. Cette dernière peut cependant se révéler délicate lorsque l'on s'intéresse à des sommets dont le voisinage n'est pas connexe. En fonction des applications visées, on utilise donc d'autres méthodes, par exemple la mémorisation directe pour chaque sommet des tétraèdres qui lui sont concourants [Frey, 1993; Verma and Gelder, 1998].

Sommets et tétraèdres peuvent être mémorisés dans des tableaux mais on leur préfère généralement des listes chaînées [Frey, 1993] qui facilitent la création de nouveaux objets.

2.5.2 Exemple de structure existante : ITK

ITK (*Insight ToolKit*) [Ibañez *et al.*, 2003] est le nom d'une bibliothèque de classes et de méthodes de segmentation d'images écrite en `C++`. En plus d'un grand nombre de filtres et de méthodes opérant directement sur les images, cette bibliothèque propose une structure très générale de maillage dont le but est de servir de support aux calculs de déformations d'une image donnée.

La caractéristique principale de la structure de donnée d'ITK est sa généralité, la même structure pouvant être utilisé pour des maillages de dimension 1 (courbes), 2 (surface), 3 (volume) ou même de dimension n (maillages généralisés). Un maillage se divise en trois partie : une partie *topologique*, une partie *géométrique*, et une partie de *données*, cette dernière pouvant être configurée en fonction de l'application désirée. La partie topologique se compose d'un certain nombre d'objets appelés *cellules*. Une cellule peut correspondre à un sommet (cellule de dimension 0), à une ligne ou à une arête (cellule de dimension 1), à un triangle ou à une facette (cellule de dimension 2), à un tétraèdre, un hexaèdre, ou un polytope quelconque (cellule de dimension 3), ou même, pour les maillages généralisés, à des structures de dimension supérieures. Chaque cellule pointe sur un certain nombre de *points* qui apportent l'information géométrique. La géométrie est donc totalement déconnecté de la topologie. Par exemple un sommet (cellule de dimension 0) ne contient pas directement d'information géométrique mais pointe sur un point. En fait, on pourrait imaginer un maillage purement topologique et ne contenant aucune information géométrique. Les relations d'adjacences ne sont ni induites, ni même réflexives, et doivent être énoncées exhaustivement.

Pour plus d'information sur la structure de donnée de maillage utilisée par ITK on pourra se reporter au chapitre 4.3 du manuel de la bibliothèque [Ibañez *et al.*, 2003].

2.5.3 Exemple de structure existante : CGAL

CGAL (*Computational Geometry Algorithms Library* [CGAL, 2002]) est le nom d'une bibliothèque de classes et de méthodes de géométrie algorithmique écrite en `C++`. Entre autres choses, cette bibliothèque propose des structures de données pour les maillages volumiques et surfaciques qui ont inspiré la nôtre. L'idée est de rendre les triangulations en dimension n homéomorphes au pavage de la sphère de dimension $n+1$. Pour cela les maillages sont clos à l'aide d'un sommet *infini* à peu près équivalent au sommet virtuel présenté précédemment.

Par souci d'économie, seuls sont explicités les sommets et les tétraèdres : les notions d'arêtes et de triangles sont donc implicites. C'est en fait la principale raison pour laquelle on a choisi de na pas utiliser cette bibliothèque et de développer une structure de données particulière. Les tétraèdres possèdent un pointeur vers chacun de leur quatre sommets ainsi que vers leurs quatre tétraèdres voisins. Les sommets ne possèdent de pointeurs que vers un seul de leurs tétraèdres concourants. La détermination des voisinages est cependant possible du fait de la clôture topologique

du maillage et des garanties de connexité qu'elle entraîne. CGAL propose aussi des objets appelés *itérateurs* et *circulateurs*. Ces objets permettent de faciliter l'accès au voisinage d'un sommet ou d'une arête donnée en automatisant son calcul et son parcours. On réutilisera le concept d'itérateur dans notre structure de données (cf. § 2.6.2). Pour plus d'information sur la structure de données des maillages de CGAL on peut se reporter à [Teillaud, 1999] ou plus directement aux chapitres 18 et 19 du manuel de la bibliothèque [CGAL, 2002].

2.6 Structure utilisée

2.6.1 Généralités

On a choisi de créer explicitement un objet pour chacune des primitives du maillage : sommets, arêtes, faces et tétraèdres. Ce choix est guidé par la volonté de faciliter l'écriture de nouveaux algorithmes de déformation, algorithmes pouvant éventuellement intégrer les notions d'arêtes et de triangles. L'explicitation des arêtes du maillage, et dans une moindre mesure celle des triangles, est source de nombreuses complications et de redondances qui ont parfois amené les programmeurs à préférer intégrer les informations dont sont porteurs ces objets à l'intérieur même des tétraèdres. L'un des problèmes principaux créés par cette introduction est celui de la redondance. Lors de la création ou de la modification du maillage, il faut en effet pouvoir déterminer facilement s'il est nécessaire de créer l'arête (ou le triangle) reliant deux (ou trois) points donnés et, si elle existe déjà, pouvoir la retrouver. Ces informations ne sont généralement disponibles qu'au prix de la construction et du parcours du voisinage des sommets concernés. Pour remédier à ce problème, on a indexé les arêtes et les triangles en fonction de leurs sommets (cf. § 2.6.8 la description de l'objet `Tetra3D`).

En plus des primitives géométriques, on a créé deux types d'objets appelés *zone* et *zone de surface*. Ces objets sont des collections, de tétraèdres pour le premier, de triangles pour le second. Chaque tétraèdre (resp. triangle) appartient au plus à une zone (resp. zone de surface). Les zones de surface permettent l'utilisation de différentes textures sur le maillage en mémorisant certaines informations relatives aux sommets telles que les coordonnées de texture. Les zones de tétraèdres n'ont pas, elles, d'utilisation prédéfinies mais peuvent être utilisées pour différencier les sous-structures du maillage et éventuellement de donner des comportements différents à chacune d'elles. Par exemple, on peut imposer une rigidité plus forte aux tumeurs lors d'un exercice de palpation [Picinbono, 2001, chapitre 5] ou encore utiliser des schémas de résolution distincts pour différentes parties du maillage [Checoury, 2002].

Chaque objet composant le maillage (sommet, arête, triangle, etc.) possède un indice de référence qui l'identifie de manière unique parmi les objets de même type. Le premier intérêt d'un tel indice est de faciliter les interactions avec l'utilisateur. Les mécanismes de sélection d'OpenGL [Woo *et al.*, 1997, chapitre 13] sont en effet basés sur une indexation des primitives par un entier de référence. Ces mécanismes sont

utilisés d'une part pour la technique de détection de collision LCN décrite plus loin (cf. § 4.4.1), mais aussi lors de la sélection manuelle de primitives à l'aide la souris. Un deuxième intérêt de cette indexation est qu'en empêchant le réemploi d'indices déjà utilisés, on dispose d'un moyen simple et efficace permettant de savoir si un objet donné a été détruit ou non. On n'a alors plus besoin de mécanismes plus lourds tels que les *pointeurs intelligents* détectant automatiquement la destruction de l'objet référencé [Ibañez *et al.*, 2003]. La question de la validité d'un pointeur donné apparaît lorsque, entre l'instant où on récupère l'adresse d'un objet et celui où on l'utilise, il a pu se produire des modifications ayant entraîné la destruction de cet objet. Cela arrive par exemple lorsque un outil détruit les tétraèdres correspondant à la matière saisie par un autre outil.

2.6.2 Sommets

```
class TetraVertex3D {
  Tetra3D* tetrahedrisation;      – Pointeur sur le maillage
  int ref;                        – Référence unique du sommet
  bool empty;                     – Caractère réel ou virtuel du sommet
  Vec3<double> position;          – Position
  TetraTetrahedron3D*             – Pointeur sur l'un des tétraèdres con-
                    tetrahedron;    courants
  std::set<TetraTetrahedron3D*>
                tetrahedronList; – Collection des tétraèdres concourants
  TetraEdge3D* vedge;
};                                – Pointeur sur l'éventuelle arête virtuelle con-
                                    courante
```

Le pointeur sur l'arête virtuelle permet de d'optimiser la détermination du voisinage du sommet dans la surface. Il facilite aussi la distinction entre sommets de la surface et sommets intérieurs, sa valeur étant égale à `NULL` pour ces derniers.

Lors de la construction du maillage, le sommet ne pointe que sur un seul de ses tétraèdres concourants. Son voisinage étant connexe, il peut être reconstruit de proche en proche à partir de cet unique tétraèdre ; il est alors mémorisé dans la *collection* STL `tetrahedronList`. Pour plus d'information sur la bibliothèque STL (Standard Template Library), on pourra se reporter à [Stepanov and Lee, 1994]. En cas de modification topologique intervenant sur l'un des tétraèdres de ce voisinage, cette collection est simplement remise à zéro et, au besoin, devra être recalculée. Il est en fait possible de construire plusieurs types de voisinage pour un sommet donné : l'ensemble des arêtes concourantes, celui des triangles concourants ou opposés, etc. Pour plus d'efficacité, on peut grouper la construction de plusieurs de ces voisinages. Ils sont eux aussi remis à zéro en cas de modification topologique intervenant sur l'un des tétraèdres concourants.

Pour opérer sur ces voisinages, on utilise un objet appelé *itérateur*. Il existe un type d'itérateur par type de voisinage : `TetrahedronVertexIterator` pour le parcours des

tétraèdres du voisinage, `EdgeVertexIterator` pour le parcours des arêtes concourantes, `TriangleVertexIterator` pour le parcours des triangles concourants, etc. À leur création, et si cela n'a pas déjà été fait, ces objets calculent les voisinages du sommet correspondant. La surcharge des opérateurs d'incrémentation ($++$ et $--$) et d'instanciation ($*$) permet alors de parcourir facilement ces voisinages. Pour illustrer ces propos, on peut montrer un code minimal de parcours du voisinage d'un sommet :

```
TetraVertex3D* v;                          – Déclaration du sommet
...
TetrahedronVertexIterator Itt(v);          – Construction de l'itérateur sur les tétra-
                                             èdres voisins
for(;!Itt.isAtEnd();++Itt){                – Parcours du voisinage
  TetraTetrahedron3D* t = *Itt;            – Instanciation du tétraèdre voisin
  ...                                      – Opérations sur ce tétraèdre
}
```

2.6.3 Arêtes

```
class TetraEdge3D {
    Tetra3D* tetrahedrisation;     – Pointeur sur le maillage
    int ref;                       – Référence unique de l'arête
    TetraVertex3D* vertex[2];      – Pointeurs sur les sommets de l'arête
    TetraTriangle3D* triangle;     – Pointeur sur l'un des triangles adjacents
    TetraTriangle3D* vtriangle;    – Pointeur sur l'éventuel triangle virtuel ad-
};                                   jacent
```

Par convention, si l'arête est virtuelle, alors le sommet virtuel est le premier des deux. Ceci permet d'accélérer la construction et le parcours du voisinage des sommets virtuels. Le pointeur sur l'éventuel triangle virtuel adjacent permet d'accélérer le parcours du voisinage de l'arête sur la surface et de faciliter la distinction entre arête intérieure et arête de la surface.

L'arête ne pointe que sur l'un de ses triangles adjacents. La connexité et la fermeture de son voisinage en permettent en effet le parcours d'une manière simple. Comme pour les sommets, il existe des itérateurs permettant de faciliter le parcours du voisinage des arêtes, la différence principale étant qu'ici il n'y a pas nécessité de construire ces voisinages. Leur usage est identique. Notons que la bibliothèque CGAL donne le nom de *circulateurs* à de tels objets.

2.6.4 Triangles

```
class TetraTriangle3D {
    Tetra3D* tetrahedrisation;  – Pointeur sur le maillage
    int ref;                    – Référence unique du triangle
    TetraTetrahedron3D*         – Pointeurs sur les tétraèdres adjacents
              tetrahedron[2];
    TetraVertex3D* vertex[3];   – Pointeurs sur les sommets du triangle
    TetraSurfaceZone3D* szone;  – Pointeur sur l'éventuelle zone de surface
};
```

Un triangle pointe sur chacun de ses trois sommets ainsi que sur ses deux tétraèdres adjacents. Par convention, le premier des deux tétraèdres est celui correspondant à une adjacence directe au sens de la définition 2.3.8. De plus, si le triangle est virtuel, c'est le premier des trois pointeurs sur sommets qui correspond au sommet virtuel.

2.6.5 Tétraèdres

```
class TetraTetrahedron3D {
    Tetra3D* tetrahedrisation;  – Pointeur sur le maillage
    int ref;                    – Référence unique du tétraèdre
    bool empty;                 – Caractère réel ou virtuel du tétraèdre
    TetraVertex3D* vertex[4];   – Pointeurs sur les sommets adjacents
    TetraEdge3D* edge[6];       – Pointeurs sur les arêtes adjacentes
    TetraTriangle3D*            – Pointeurs sur les triangles adjacents
                 triangle[4];
    TetraTetrahedron3D*         – Pointeurs sur les quatre tétraèdres voisins
                neighbors[4];
    TetraZone3D* zone;          – Pointeur sur l'éventuelle zone de tétraèdres
};
```

Un tétraèdre possède des pointeurs sur chacun de ses quatre sommets ainsi que sur l'ensemble des six arêtes, quatre triangles et deux tétraèdres qui lui sont adjacents. Par convention, si le tétraèdre est virtuel, c'est le premier des quatre pointeurs sur sommets qui correspond au sommet virtuel. De plus on impose que le $i^{ème}$ triangle soit le triangle opposé au sommet de même indice i.

On pourrait juger inutilement redondant l'ajout de pointeurs sur les tétraèdres voisins alors que cette information est disponible dans les triangles. En fait, la préoccupation principale de notre implémentation est d'optimiser l'accès aux voisinages des objets. La taille de la structure de données n'est pas une donnée critique car les maillages utilisés pour l'ensemble de nos applications ne dépassent que rarement les 10.000 tétraèdres. L'existence de ces pointeurs permet l'accès direct aux tétraèdres voisins. Sans eux on devrait passer par un algorithme du type :

```
Si tétraèdre->triangle[i]->tétraèdre[0]==tétraèdre
```

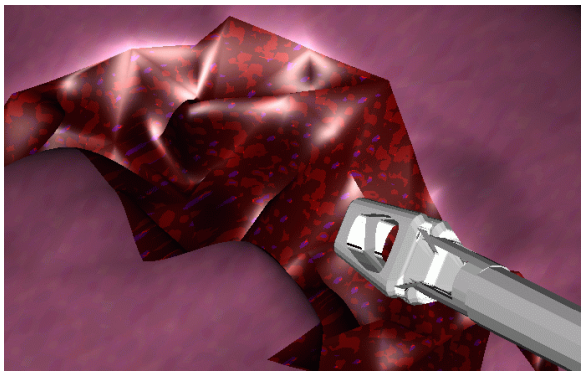


FIG. 2.8 – *L'utilisation des zones de surface permet de disposer de rendus différents pour la capsule de Glisson et pour le parenchyme hépatique.*

```
   voisin[i] = tétraèdre->triangle[i]->tétraèdre[1]
Sinon
   voisin[i] = tétraèdre->triangle[i]->tétraèdre[0]
```

2.6.6 Zones de surface

```
class TetraSurfaceZone3D {
    Tetra3D* tetrahedrisation;      – Pointeur sur le maillage
    int ref;                        – Référence unique du tétraèdre
    std::set<TetraTriangle3D*>      – Ensemble des triangles de la zone
                    triangleList;
    std::map<TetraVertex3D*,        – Informations relatives aux sommets de la
            VertexInformation*>       zone de surface
            vertexInformationMap;
};
```

```
class TetraSurfaceZone3D::VertexInformation {
    TetraVertex3D* vertex;          – Pointeur sur le sommet
    Vec3<double> normal;            – Normale au sommet
    float textureCoordinate[4];     – Coordonnées de textures
    unsigned int counter;           – Nombre de triangles de la zone qui sont
};                                    concourants au sommet (permet de suppri-
                                      mer les informations relatives à des som-
                                      mets qui ne font plus partie de la zone de
                                      surface)
```

La notion de zone de surface a dû être introduite quand on a voulu utiliser différents rendus texturés sur un même maillage. Lors de la découpe du foie par exemple, il est souhaitable de pouvoir représenter différemment les tissus correspondant à la capsule de Glisson et ceux correspondant au parenchyme hépatique (voir figure 2.8). Le problème qui se pose alors est que les sommets situés à l'interface entre des zones

de rendu différent possèdent a priori des coordonnées de textures différentes pour chacune de ces zones. Certaines applications de rendu graphique résolvent ce problème en scindant le maillage de façon à ce qu'un sommet donné n'appartienne qu'à une seule zone de rendu. La notion de zone de surface applique indirectement cette idée.

Une zone de surface est en fait l'union d'une collection de triangles sujets au même traitement graphique et d'une collection d'informations graphiques sur les sommets de ces triangles (coordonnées de texture et normale). Elle possède de plus des informations sur le traitement à appliquer aux triangles : usage et nature de la texture, définition du matériel, etc. On peut aussi leur trouver d'autres applications et les utiliser entre autres comme des objets permettant d'appliquer des conditions aux limites particulières à une partie de la surface, par exemple des forces de pression différentes pour les parois internes et externes.

2.6.7 Zones

```
class TetraZone3D {
    Tetra3D* tetrahedrisation;
    int ref;
    std::set<TetraTetrahedron3D*>
                  tetrahedronList;
  TetraSurfaceZone3D*
              cuttingSurfaceZone;
};
```

- Pointeur sur le maillage
- Référence unique de la zone
- Ensemble des tétraèdres appartenant à la zone
- Zone de surface affectée aux nouveaux triangles de surface issus de la découpe qui sont adjacents à un tétraèdre de la zone.

Une zone est une collection de tétraèdres, chaque tétraèdre réel ne pouvant appartenir au plus qu'à une zone.

La zone de surface désignée par le pointeur `cuttingSurfaceZone` est affectée par défaut aux nouveaux triangles de surface adjacents à un tétraèdre de la zone lors d'un processus de découpe. Cela permet par exemple de disposer d'une texture particulière pour l'intérieur du parenchyme, des tumeurs, etc.

2.6.8 Tétraédrisation

```
class Tetra3D {
  std::vector<????*> tetrahedronVector,
            triangleVector, edgeVector,
              vertexVector, zoneVector,
                     surfaceZoneVector;
  std::set<???*> vertexList, zoneList,
       surfaceZoneList,tetrahedronList;
  std::map<Key,???*> triangleMap,
                     edgeMap;

  std::set<TetraVertex3D *>
                     virtualVertexList;
};
```

– Vecteurs STL pointant sur les sommets, arêtes, triangles, tétraèdres, zones et zones de surface du maillage,

– Collections regroupant les sommets et les tétraèdres,

– Cartes indexant les arêtes et les triangles en fonction de leurs sommets.

– Liste des sommets virtuels,

Cet objet regroupe l'ensemble des primitives composant le maillage. Les tableaux (`*Vector`) permettent un accès aux primitives à partir de leur référence en temps constant. Ceci est particulièrement utile lors des détections de collision avec l'outil virtuel (cf. § 4.4.1).

Les cartes et les collections (`*Map` et `*List`) permettent de parcourir efficacement l'ensemble des primitives d'un type donné. Ce parcours se fait à l'aide d'itérateurs semblables à ceux décrits plus haut (cf. § 2.6.2). Arêtes et triangles ont été rangés dans des cartes et indexés par une clé construite à partir de leurs sommets. Il s'agit en fait d'un vecteur dont les éléments sont les pointeurs sur ces sommets. Ce dispositif permet de répondre efficacement aux interrogations sur l'existence d'une arête ou d'un triangle reliant deux ou trois sommets donnés (cf. § 2.6.1), par exemple lors de la création ou de la modification du maillage (cf. § 2.7)

2.7 Construction et modification du maillage

L'expérience montre qu'il peut se révéler très laborieux de mettre en œuvre les fonctions modifiant la topologie du maillage, et cela même si les opérations correspondantes sont relativement simples. Il faut non seulement construire les tétraèdres de manière correcte, mais aussi mettre à jour tous les champs et les informations d'adjacence. Même si ce travail ne doit être effectué qu'une fois par type d'opération, lors de l'écriture de la fonctionnalité, cela peut se révéler limitant, rendant éventuellement hors de portée la réalisation de ces opérations. On cherche donc à décharger le plus possible le programmeur et à réduire son travail au simple listage des tétraèdres orientés qui composeront son maillage.

2.7.1 Création des primitives

Pour rendre transparents les mécanismes d'héritage, il importe de définir clairement la manière avec laquelle peuvent être créées les primitives. L'idée est de définir un prototype de créateur qui devra être respecté par tous les objets dérivés d'un même type (sommet, arête, etc.). Les initialisations particulières pourront alors être effectuées de trois manières différentes :

- soit directement à l'intérieur du constructeur de l'objet (valeurs par défaut) ;
- soit, dans le cas de la création du maillage, explicitement et à la fin de cette création ;
- soit, dans le cas de la modification d'un maillage existant, à l'aide d'un mécanisme expliqué plus loin.

La création du maillage revient alors, pour le programmeur :

- premièrement à créer les sommets du maillage à partir des positions désirées ;
- deuxièmement à créer les tétraèdres à partir de leurs sommets, en veillant à bien les orienter correctement.

Les triangles et les arêtes sont créés automatiquement selon les besoins. La création de sommets et des tétraèdres virtuels peut, elle aussi, être automatisée en considérant les triangles n'ayant qu'un seul tétraèdre voisin.

Notons que la détermination automatique de l'orientation du tétraèdre serait théoriquement possible à l'aide de simples considérations géométriques (signe de $\vec{AB} \wedge \vec{AC}.\vec{AD}$). Malheureusement, ces considérations ne sont pas valables pour les tétraèdres virtuels. De plus, un maillage déformé peut présenter des retournements de tétraèdres mettant eux aussi ces considérations en défaut. On n'a donc pas d'autre choix que d'imposer à l'utilisateur de préciser l'orientation de ses nouveaux tétraèdres.

2.7.2 Modification du maillage

La modification de maillage est un processus qui consiste à retirer un certain nombre de tétraèdres du maillage et à les remplacer par d'autres. Cette opération doit généralement s'accompagner d'une mise à jour de certains champs ou encore de l'accomplissement de certaines opérations. Par exemple, pour un maillage masse-ressort, il faudra initialiser les longueurs au repos des nouvelles arêtes ; pour un maillage masse-tenseur, il faudra mettre à jour les valeurs des tenseurs d'élasticité, les positions de repos, etc. ; pour un maillage hybride masse-tenseur/précalculé, il faudra éventuellement faire basculer une zone du mode précalculé au mode masse-tenseur, etc. Le problème que posent ces mises à jour est que le développeur des opérations topologiques n'a pas forcement connaissance de leur nature exacte. Ceci est particulièrement vrai lorsque ces modifications sont appliquées sur une classe de maillage dérivée de la classe de base. Le développeur ne dispose alors que d'une information partielle sur la nature des objets manipulés. Il est donc nécessaire de prévoir un mécanisme permettant d'effectuer ces mises à jour de manière transparente.

Pour résoudre ces problèmes, on utilise le mécanisme de fonction virtuelle du langage `C++`. Ce mécanisme permet au développeur d'une classe dérivant d'une classe de base, de redéfinir certaines des méthodes de cette classe de base. Le développeur des opérations topologiques appelle des fonctions génériques censées résoudre les problèmes de mise à jour particuliers, charge au développeur des maillages hérités d'implémenter correctement ces fonctions. On fournit le prototype de trois fonctions :

- `removeFromMesh()` (fonction membre des classes de tétraèdres) : cette fonction résout les problèmes liés au retrait d'un tétraèdre du maillage, par exemple soustraire aux tenseurs d´élasticité voisins les contributions de ce tétraèdre ou basculer la zone correspondante du modèle précalculé au modèle masse-tenseur.
- `setProperties()` (fonction membre des classes de sommets) : cette fonction prend en argument de 1 à 4 pointeurs sur des sommets affectés de 1 à 4 coefficients réels. Elle initialise les champs du sommet en fonction des 1 à 4 sommets fournis en argument et en proportion des coefficients correspondants. La position est l'un des différents champs mis à jour par cette fonction. Pour les maillages hérités, on peut initialiser aussi des grandeurs telles que la position au repos, la position précédente ou encore l'orientation de l'anisotropie locale.
- `setProperties()` (fonction membre des classes de tétraèdres) : cette fonction prend en argument un tétraèdre et prend en charge les mises à jours dues à l'insertion du tétraèdre dans le maillage. Pour un tétraèdre virtuel, elle met à jour la zone de surface du triangle réel correspondant. Pour un tétraèdre réel, elle initialise les valeurs des coefficients de Lamé λ et μ, et met à jour les tenseurs d'élasticité.

Bien sûr, pour un maillage hérité, ces fonctions doivent appeler les fonctions des classes de base ou alors prendre à leur charge les mises à jour correspondantes.

Les processus de modification se déroulent donc toujours de la même façon :

1. détermination de la liste des tétraèdres qui seront retirés du maillage,
2. application sur chacun de ces tétraèdres de la fonction appelée `removeFromMesh()`,
3. construction des nouveaux sommets et mise à jour de leurs propriétés à l'aide de la fonction `setProperties()`,
4. construction des nouveaux tétraèdres,
5. initialisation des nouveaux tétraèdres à l'aide de la fonction `setProperties()`,
6. destruction des tétraèdres, triangles arêtes et sommets retirés du maillage,
7. éventuelle division du point virtuel (cf. § 2.7.3)

L'opération finale consiste à déterminer puis à détruire les objets retirés du maillage. Cette détermination peut se révéler laborieuse : en effet un sommet ou une arête ne sont détruits que si la totalité des tétraèdres concourants ou adjacents sont retirés du maillage. Cette opération peut cependant être facilement automatisée en utilisant la redondance de la structure de données. En effet, lors de sa création, un tétraèdre met automatiquement à jour les pointeurs des sommets arêtes et triangles qui

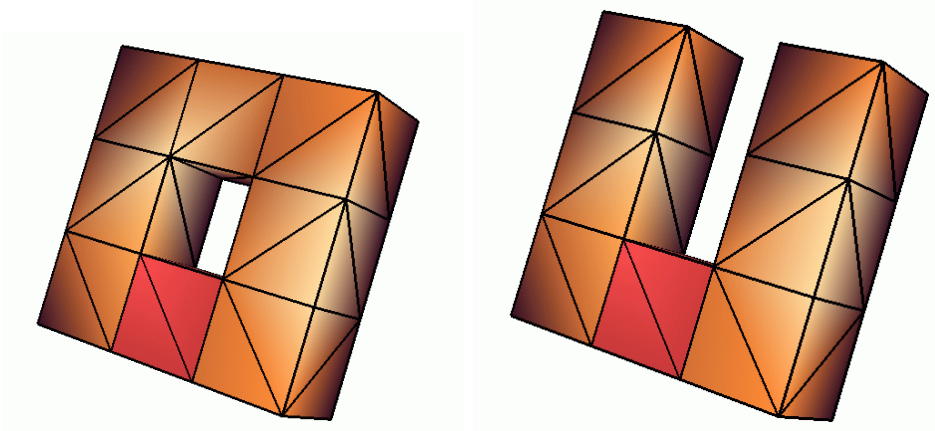


FIG. 2.9 – *La suppression des tétraèdres en rouge crée une nouvelle composante connexe à droite mais pas à gauche, alors que localement elles sont identiques.*

lui sont adjacents. Les sommets à retirer du maillage sont donc, parmi les sommets concernés par la modification, ceux pointant toujours vers l'un des tétraèdres supprimés. On peut opérer de même pour les triangles, puis pour les arêtes en vérifiant qu'elles ne pointent pas sur l'un des triangles supprimés.

2.7.3 Gestion des point virtuels

Les points virtuels posent un problème particulier lors de l'écriture d'opérations topologiques. On a en effet vu précédemment (cf. observation 2.3.16) que l'on doit avoir un sommet virtuel par composante connexe. La scission d'un sommet virtuel entre ses deux composantes connexes est une opération assez coûteuse, surtout pour des grandes surfaces, car il faut premièrement identifier exactement chacune des composantes connexes, et deuxièmement, dans chaque nouvelle composante, remplacer chaque pointeur sur le sommet virtuel par un pointeur sur un nouveau sommet virtuel. Pour éviter des calculs inutiles, il faudra donc essayer de déterminer la nécessité de la création d'un nouveau sommet virtuel. Il est cependant quelquefois difficile de savoir s'il faut ou non créer un sommet virtuel sans avoir à considérer l'ensemble du maillage (cf. figure 2.9).

2.8 Tests topologiques

Il est important d'être capable de vérifier la validité d'un maillage stocké en mémoire, c'est-à-dire d'une part de vérifier l'intégrité de la structure de données, et d'autre part de vérifier que cette structure de données correspond bien à un maillage topologiquement clos au sens de la définition 2.3.13. Cette vérification est une opération assez lourde et s'effectue en plusieurs étapes.

La première étape consiste à vérifier la cohérence de la structure de données. Il

s'agit de vérifier que les sommets pointent bien vers l'un de leurs tétraèdres concourants, les arêtes sur l'un de leurs triangles adjacents, les triangles sur leurs deux tétraèdres adjacents et les tétraèdres sur les arêtes, triangles et voisins correspondants. Il faut de plus vérifier l'ordonnancement des différents tableaux de pointeurs :

1. sommet virtuel toujours en première position pour une arête, un triangle ou un sommet,
2. tétraèdre en adjacence directe toujours le premier des deux pour un triangle,
3. position relative des triangles et des sommets pour les tétraèdres,
4. non existence de doublons en regardant dans les cartes,
5. valeur du pointeur sur l'arête virtuelle concourante pour les sommets ou sur le triangle adjacent pour les arêtes.

Ces opérations peuvent être effectuées par un simple parcours des structures et fournissent déjà un certain nombre d'indications sur la validité du maillage. En fait, on a déjà la vérification des hypothèses H3, H4 et H5 de la définition 2.3.13. De plus, si on ne considère pas comme critique la propriété géométrique de non intersection de deux tétraèdres, on a aussi la vérification de l'hypothèse H2. En fait, seule reste à vérifier l'hypothèse H1, c'est-à-dire que le maillage est bien un maillage topologique (cf. définition 2.3.11) ; et dans cette définition seul un point pose problème : il s'agit de vérifier la connexité du voisinage des sommets et des arêtes.

Cette opération est plus laborieuse car la seule manière de vérifier cette connexité est de calculer l'ensemble des voisinages (à l'aide des itérateurs présentés en 2.6.2 et 2.6.3), puis de vérifier que tous les tétraèdres appartiennent bien aux voisinages des sommets et arêtes qui leurs sont adjacents. La vérification de cette connexité est cependant très importante car elle permet l'utilisation des outils très efficaces que sont les itérateurs.

2.9 Hiérarchie de maillages

On va présenter succinctement différents types de maillage déjà présents dans `yav+++` et héritant de la classe de base décrite plus haut.

2.9.1 ActiveTetra3D

Cette classe a été développée pour permettre la déformation des maillages. Trois nouveaux objets sont proposés :

– `InternalForce3D` Cet objet est chargé du calcul des forces internes. Il existe un objet par type de force (masse ressort, masse-tenseur linéaire, masse-tenseur non linéaire, néo hookéen, etc.). Chaque maillage pointe vers l'un de ces objets et exécute régulièrement une méthode nommée `computeInternalForce()` chargée de calculer les forces internes de la manière désirée. Si on veut qu'elle supporte

les modifications topologiques, une force doit fournir en plus deux fonctions permettant la mise à jour des grandeurs mécaniques qu'elle utilise lors du retrait et de l'ajout d'un tétraèdre.

– `ForceConstrainte3D` Cet objet est chargé du calcul des forces externes encore appelées *contraintes de force*. Il existe un objet par type de contrainte de force. Chaque maillage peut posséder plusieurs de ces objets et exécute régulièrement une fonction appelée `apply()` qui va calculer, de la manière désirée, des forces supplémentaires appliquées au maillage. Il existe un grand nombre de contraintes de force disponibles (contrainte d'incompressibilité, force de pression, contrainte d'isovolumétrie, etc.) et chaque bibliothèque en propose de nouvelles.
– `PositionConstraint3D` Cet objet, appelé *contrainte de position* est semblable au précédent à la différence qu'au lieu de calculer des forces, il impose des déplacements.

La classe `ActiveTetra3D` dispose d'une fonction dénommée `iterate()` qui est appelée régulièrement et qui est chargée du calcul et de l'application des déformations. Pour simplifier, on peut considérer que cette fonction procède de la manière suivante :

1. calcul des forces externes générées par les contraintes de force,
2. calcul des forces internes,
3. mise à jour des positions,
4. déplacements générés par les contraintes externes.

L'intérêt de ce dispositif est sa grande modularité. Il est en effet possible de créer à volonté de nouvelles forces ou de nouvelles contraintes et même d'en changer en cours d'utilisation.

2.9.2 SimuTetra3D

Il s'agit du maillage utilisé dans le cadre du simulateur. Il permet de placer des vaisseaux à l'intérieur du maillage. Ces vaisseaux sont liés au maillage par un mécanisme force/position qui sera expliqué en détail plus loin (cf. § 5.3.4). Cette classe de maillage hérite de la classe de maillages déformables `ActiveTetra3`.

La fonction `iterate` a été modifiée de façon à contrôler la déformation des vaisseaux et permettre à ceux-ci de transmettre leurs forces internes au maillage.

2.9.3 HeartTetra3D

Cette classe de maillage permet le calcul des potentiels d'activation électriques du cœur et celui des déformations engendrées. Les sommets possèdent une grandeur supplémentaire appelée *potentiel d'activation* ainsi qu'une *direction d'anisotropie* correspondant à l'orientation locale des fibres. De nouvelles contraintes sont proposées qui permettent de calculer la propagation de l'activation ou d'appliquer une force de pression sur certaines des zones de surface du maillage. Des mécanismes ont de plus

été mis en place pour visualiser les potentiels sous forme de texture 1D. Pour plus d'information sur ces travaux, on pourra se reporter à la thèse de Maxime Sermesant [Sermesant, 2003].

2.9.4 HybridMesh

Cette classe met en œuvre le modèle mixte précalculé masse-tenseur proposé par Stéphane Cotin [Cotin, 1997, chapitre 6]. Le maillage est composé de plusieurs zones utilisant un modèle de déformation précalculé. Ce modèle de déformation est très rapide mais ne permet pas de procéder à des changements topologiques. Lorsque l'utilisateur veut procéder à une découpe, la zone concernée passe automatiquement en masse-tenseur. Les calculs de déformation sont alors plus lents mais la découpe devient possible. Pour plus d'information sur ces travaux, on pourra se reporter au rapport de DEA de Cédric Checoury [Checoury, 2002].

2.9.5 Brain

La librairie `libBrain` a été écrite pour permettre une modélisation des déformations de cortex [Clatz, 2002].

2.10 Conclusion

Nous avons proposé une nouvelle structure de données pour des maillages volumiques ou surfaciques pouvant servir de support à un grand nombre d'applications basées sur les calculs de déformation et la visualisation. L'existence des itérateurs, qui permettent une automatisation de la détermination et du parcours des voisinages, permet d'optimiser les méthodes de calcul de déformation et de faciliter leur écriture. La création explicite d'un objet pour chaque primitive du maillage (sommets, arêtes, triangles et tétraèdres) et le système modulaire de forces et de contraintes proposé pour le système déformable lui permettent d'être également utilisée dans un grand nombre d'applications.

En contrepartie, cette structure présente un certain nombre de limitations. La création explicite d'un objet pour chaque primitive du maillage et l'existence d'objets virtuels en font une structure relativement gourmande en espace mémoire. Elle n'est donc pas adaptée à la manipulation de maillages de trop grande taille (ie $>$ 10.000 tétraèdres). L'existence des objets virtuels compliquent aussi sensiblement les opérations de modification topologiques du maillage par rapport aux structures de données basées sur une simple accumulation d'objets.

Il reste un certain nombre d'améliorations à effectuer à cette structure. Il faudrait par exemple améliorer les mécanismes de mise à jour des propriété utilisées lors des modifications topologiques. La technique actuelle ne permet en effet que des

raffinements par subdivision. Faire dépendre des zones, et non plus des tétraèdres, la détermination des propriétés permettrait d'envisager des méthodes de raffinement plus subtiles telles que Delaunay par exemple (cf. § 3.11). Il faudrait aussi extraire des objets de base du maillage la totalité des données spécifiques au calcul des forces (valeur des coefficients λ et μ, direction d'anisotropie, etc.) et mettre en place un système entièrement géré par les forces, ceci afin de purger le modèle de maillage de la quantité de valeurs n'étant utiles que dans certains cas.

Chapitre 3

Découpe

Sommaire

3.1 Importance de la découpe

La découpe des maillages est un domaine de recherche encore très actif. Les problématiques sont nombreuses : modélisation de fractures [Norton *et al.*, 1991; O'Brien and Hodgins, 2000], de déchirements [Boux de Casson, 2000; Boux de Casson and Laugier, 2000], animation ou simulation, temps réel ou non, etc., et une approche applicable pour un problème donné peut ne pas l'être pour d'autres. La découpe de maillages volumiques dans le cadre d'une simulation temps réel soulève plusieurs problèmes particuliers. D'abord, les modifications effectuées sur le maillage, lors de cette découpe, ne doivent pas gêner la poursuite de la simulation. Il faut donc limiter au maximum la quantité de tétraèdres créés afin de ne pas causer un ralentissement excessif de la simulation. Il ne faut pas non plus que les nouveaux tétraèdres soient ni trop petits, ni de qualité trop médiocre, car cela peut entraîner des instabilités dans le calcul des déformations. De plus, et comme nous le verrons plus loin, il faut empêcher la création de *singularités topologiques*. Enfin, il faut rendre visuellement acceptables les maillages résultants de cette découpe. Cette dernière exigence est en contradiction avec celle d'admissibilité pour la simulation car elle tend généralement à augmenter le nombre de tétraèdres créés et à diminuer leur qualité (cf. § 3.10.2 pour une discussion sur la qualité des tétraèdres)

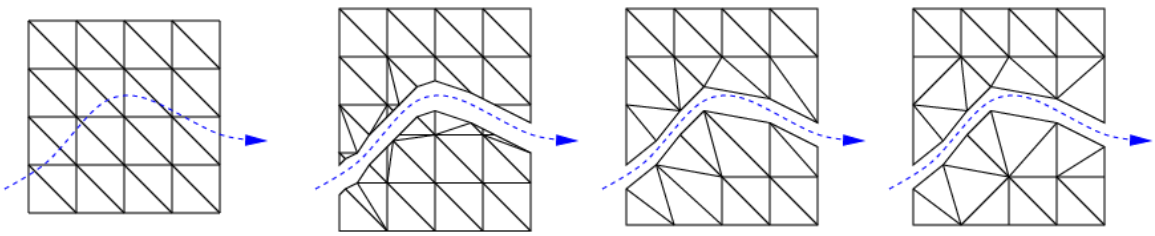


FIG. 3.1 – *Trois méthodes de découpe 2D par incision : scission de tétraèdres, déplacement de sommets, et déplacement de sommets assorti d'un remaillage de type Delaunay*

Il existe deux manières de considérer la découpe en simulation de chirurgie. La première cherche à simuler l'effet d'un bistouri : on parle de découpe *par incision*. L'idée est de modifier le maillage de façon à le scinder le long d'une surface de découpe définie par le parcours de l'outil (cf. figure 3.1). Cette scission se fait généralement à l'aide de subdivisions de tétraèdres [Bielser and Gross, 2000; Mor and Kanade, 2000]. Si ces méthodes permettent des découpes extrêmement précises, même à l'intérieur de maillages très grossiers, elles ont le défaut de créer un très grand nombre de petits tétraèdres, souvent de piètre qualité. D'autres approches proposent de modifier la position des sommets du maillage de façon à déplacer sur la surface de découpe des facettes déjà existantes, limitant ainsi la création de nouveaux tétraèdres [Boux de Casson, 2000; Nienhuys and van der Stappen, 2001]. Le défaut de telles méthodes reste une dégradation du niveau de qualité du maillage. Ont été récemment évoquées des méthodes de découpe de maillages surfaciques combinant le déplacement de sommets existants avec un remaillage local de type Delaunay [Nienhuys and van der Stappen, 2002].

L'applicabilité de telles méthodes pour les maillages volumiques n'est cependant pas encore démontrée. Un autre défaut de ce type de méthode est de supposer que les découpes s'effectuent en un seul geste, relativement régulier. Ceci n'est malheureusement que rarement le cas en chirurgie où les découpes se font plutôt par à-coups successifs. Ces méthodes sont donc encore d'usage limité en simulation de chirurgie, du moins en ce qui concerne la découpe de maillages volumiques.

Lors d'une hépatectomie, le chirurgien a le choix entre plusieurs méthodes pour procéder à la découpe. Il peut par exemple utiliser une *pince cautérisante*. Le parenchyme hépatique, très friable, est facilement détruit sous l'action de cette pince. De plus, manipulée avec précaution, elle n'endommagera pas les vaisseaux rencontrés, l'action cautérisante permettant de limiter les pertes sanguines. Il peut aussi utiliser un *bistouri à ultrasons*, encore appelé *cavitron*. L'extrémité de cet instrument émet des ultrasons qui causent la destruction des cellules hépatiques environnantes. Bien calibrées, ces ondes n'endommagent pas le réseau vasculaire, limitant ici encore les pertes sanguines. Dans les deux cas, on peut parler de découpe *par retrait de matière*. Ce type de découpe est modélisable par un simple retrait de tétraèdres, ce qui présente le double avantage de ne pas créer de nouveaux éléments et de ne pas diminuer la qualité du maillage. Si la taille des tétraèdres est suffisamment petite devant celle de l'outil, l'apparence irrégulière de la surface de découpe est assez réaliste ; inversement, la découpe d'un maillage trop grossier peut se révéler décevante, même si certains effets graphiques tels que l'utilisation d'une texture appropriée améliore l'impression visuelle. Pour éviter d'avoir à choisir entre des calculs de déformation rapides avec une découpe grossière, et des calculs de déformation lents avec une découpe précise, on utilise une méthode de raffinement local dynamique qui sera décrite plus loin (§ 3.10)

3.2 Notion de variété

Dans le cadre du simulateur, la variété du maillage nous intéresse premièrement pour les propriétés graphiques qui en découlent (rendu, lumière, etc.). Nous intéresse aussi la possibilité de calculer une force de réaction lors du contact entre l'outil et le maillage (cf. § 4.5). La variété y est nécessaire car cette réaction s'exerce selon la normale à la surface au point de contact et cette normale n'est définie que si la surface du maillage est une variété (cf. observation 2.4.7). Enfin, et pour pouvoir profiter des facilités de détermination du voisinage, la structure de données que nous utilisons nous contraint à la manipulation de solides variétés (cf. observation 2.4.6 et § 2.8).

Le problème que pose la découpe des maillages volumiques est qu'elle peut facilement conduire à des maillages présentant des singularités topologiques (cf. § 3.8). On peut par exemple montrer qu'il n'est possible ni de percer une surface, ni de séparer deux composantes connexes, par un retrait de matière tétraèdre par tétraèdre, sans passer par une configuration présentant une singularité.

Paradoxalement, il n'y a que très peu de littérature se préoccupant du maintien de la variété lors de la découpe de maillages volumiques. Certains auteurs ont bien

reporté quelques problèmes [Nienhuys and van der Stappen, 2001; Mendoza *et al.*, 2001] mais sans jamais chercher à garantir le maintien de cette propriété.

Observation 3.2.1 *Il n'est pas possible de percer un maillage (ie. de passer d'un maillage homéomorphe à une sphère à un maillage homéomorphe à un tore) en retirant les tétraèdres un à un sans passer par une configuration présentant une singularité.*

Preuve : La figure 3.2 illustre le raisonnement pour le cas 2D. Soit $(\mathcal{T}_i)_{i\in[0,n]}$ une suite de maillages volumiques vérifiant les trois points suivants :

- le maillage $\mathcal{T}_0$ est homéomorphe à une sphère,
- le maillage $\mathcal{T}_n$ est homéomorphe à un tore,
- le maillage $\mathcal{T}_{i+1}$ ne diffère du maillage $\mathcal{T}_i$ que par le retrait d'un tétraèdre que l'on nommera T_i.

Il existe évidemment de telles suites de maillages. Soit $\mathcal{T}_{j+1}$ le premier maillage pour lequel il existe un "trou". Soit $f : I\!R \to I\!R^3$ une courbe passant par ce trou et n'étant en contact avec aucun des tétraèdres de $\mathcal{T}_{j+1}$.

1. Le maillage $\mathcal{T}_j$ ne présentant pas de trou f doit l'intersecter et cette intersection se produit sur le tétraèdre T_j (unique différence entre les deux maillages).
2. Supposons que la courbe f n'intersecte que deux fois le tétraèdre T_j (et donc le maillage $\mathcal{T}_j$), c'est-à-dire qu'elle ne fait pas de détours inutiles, et soient F_1 et F_2 les deux triangles de la surface du maillage $\mathcal{T}_j$ intersectés par f. L'arête E commune à ces deux triangles fait forcement partie de la surface du maillage $\mathcal{T}_{j+1}$, sinon on pourrait faire passer la courbe f entre cette arête E et le maillage $\mathcal{T}_{j+1}$, et le maillage $\mathcal{T}_j$ présenterait un trou.
3. Les deux triangles F_3 et F_4 adjacents à E sur la surface du maillage $\mathcal{T}_{j+1}$ ne sont pas des triangles du tétraèdre T_j. En effet, si cela était le cas, ils seraient forcément égaux à F_1 et F_2. Or, ceux-ci, ne sont adjacents qu'au tétraèdre T_j et ne font donc pas partie du maillage $\mathcal{T}_{j+1}$.
4. Les deux triangles F_3 et F_4 appartiennent à la surface du maillage $\mathcal{T}_{j+1}$ et n'appartiennent pas au tétraèdre T_j : ils appartiennent donc à la surface du maillage $\mathcal{T}_j$.
5. L'arête E est donc adjacente à quatre triangles de la surface du maillage $\mathcal{T}_j$, ce maillage présente donc une singularité.

◇

Observation 3.2.2 *Il n'est pas possible de scinder un maillage (ie. passer d'une composante connexe à deux composantes connexes) en retirant les tétraèdres un à un sans passer par une configuration présentant une singularité.*

Preuve : Soit $(\mathcal{T}_i)_{i\in[0,n]}$ une suite de maillages volumiques vérifiant les trois points suivants :

- le maillage $\mathcal{T}_0$ possède une seule composante connexe,

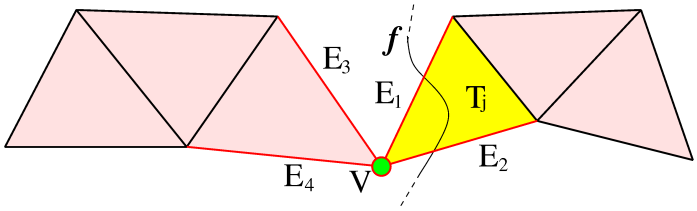


FIG. 3.2 – *Vision 2D de la preuve 3.2.1. On voit le maillage $\mathcal{T}_j$, le tétraèdre T_j et la courbe f. Le sommet V correspond à l'arête E et les arêtes E_1, E_2, E_3 et E_4 respectivement aux triangles F_1, F_2, F_3 et F_4.*

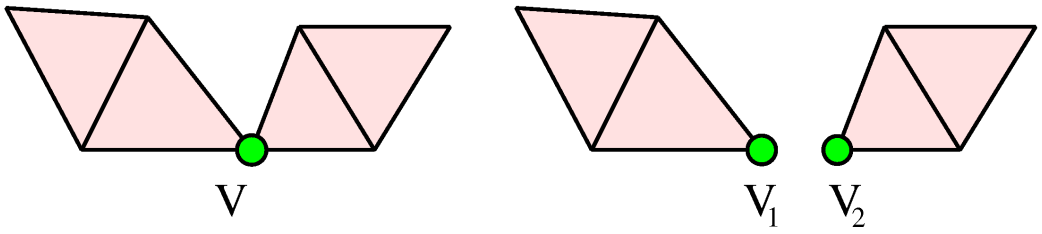


FIG. 3.3 – *Suppression de singularité en 2D*

- le maillage $\mathcal{T}_n$ possède exactement deux composantes connexes,
- le maillage $\mathcal{T}_{i+1}$ ne diffère du maillage $\mathcal{T}_i$ que par le retrait d'un tétraèdre que l'on nommera T_i.

Il existe évidemment de telles suites de maillages. Soit $\mathcal{T}_{j+1}$ le premier maillage pour lequel il existe deux composantes connexes (ie. ensemble de tétraèdres n'ayant aucun sommet commun). Soient $\mathcal{C}_1$ et $\mathcal{C}_2$ ces deux composantes connexes. Comme le maillage $\mathcal{T}_j$ ne possède qu'une composante connexe, le tétraèdre T_j doit être en contact avec les deux composantes connexes du maillage $\mathcal{T}_{j+1}$. Ces deux composantes connexes n'ayant aucun sommet commun, l'une au moins n'est en contact avec T_j que par au plus deux sommets. Supposons que cela soit $\mathcal{C}_1$. Si elle n'est en contact avec le tétraèdre T_j que par deux sommets, alors l'arête correspondante est adjacente à quatre triangles de la surface du maillage $\mathcal{T}_j$, deux triangles de $\mathcal{C}_1$ et deux triangles de T_j. Le maillage $\mathcal{T}_j$ présente donc une singularité localisée sur cette arête. Si $\mathcal{C}_1$ n'est en contact avec le tétraèdre T_j que par un seul sommet, le voisinage de ce sommet sur la surface du maillage $\mathcal{T}_j$ possède deux composantes connexes, l'une sur $\mathcal{C}_1$ et l'autre composée de triangles de T_j. Le maillage $\mathcal{T}_j$ présente donc une singularité localisée sur ce sommet. ◇

3.3 Stratégies de suppression de singularités

La suppression de singularités est une opération triviale en deux dimensions où il est possible de scinder les primitives posant problème (voir figure 3.3). Cela n'est pas le cas en trois dimensions, et la régularisation de maillages non variété est encore un

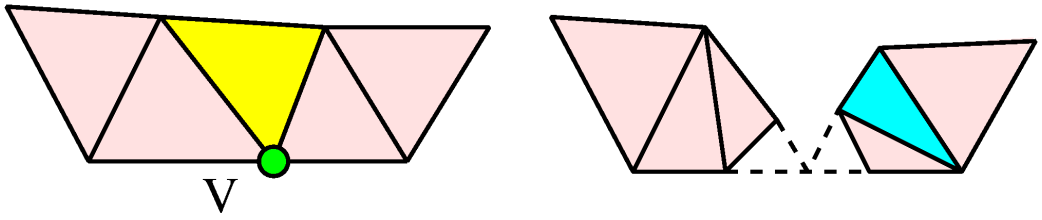


FIG. 3.4 – *Le retrait du tétraèdre jaune crée une singularité au sommet V. On peut raffiner autour de ce sommet puis retirer le voisinage. Le retrait du tétraèdre bleu créerait une nouvelle singularité*

champ actif de la recherche [Guéziec *et al.*, 2001].

Le retrait de tétraèdres dans un maillage volumique est une opération qui peut facilement créer des singularités et on peut imaginer plusieurs stratégies pour prévenir ces créations. La première procède par analogie avec la méthode efficace en 2D et consiste à scinder les singularités. Malheureusement en 3D, dans le cas d'une singularité localisée sur un sommet, la différenciation entre la partie "à droite" et la partie "à gauche" de la singularité n'est généralement pas possible. On peut s'en sortir en dédoublant tous les triangles concourants à la singularité et en en démultipliant les sommets correspondants mais cela a tendance à créer facilement de nouvelles singularités, nécessitant de nouvelles scissions. Un autre défaut de cette méthode est d'augmenter notablement les possibilités d'auto-intersections du maillage, mais cela diminue le réalisme visuel de la simulation et risque de poser des problèmes lors de l'interaction avec les outils (cf. § 4.4). Une autre stratégie consiste à raffiner localement autour des singularités puis à retirer le voisinage de celles-ci. Cette méthode est intéressante car elle fournit à coup sûr une solution. Elle a cependant le défaut de ne pouvoir garantir l'absence de singularités futures aux endroits déjà raffinés ce qui peut entraîner des raffinements en cascade (voir figure 3.4).

L'algorithme que nous proposons consiste à analyser le retrait d'un unique tétraèdre et à différencier les cas où ce tétraèdre ne peut être retiré du maillage sans créer de singularité. On résout alors spécifiquement chacun des cas rencontrés, soit, lorsque cela est possible, à l'aide d'une simple scission d'arête, soit en déterminant un ensemble de tétraèdres $\mathcal{T}_{suppr}$, contenant le tétraèdre de départ, et dont le retrait ne fait pas perdre au maillage son caractère variété. Dans certains cas, peu nombreux, notre algorithme ne fournit pas de solution. Ce dernier point sera discuté à la partie 3.8.

3.4 Retrait d'un unique tétraèdre

La première étape de notre algorithme est la classification des différentes situations que l'on peut rencontrer lorsque l'on veut retirer un tétraèdre du maillage.

Observation 3.4.1 *Soit T le tétraèdre que l'on cherche à retirer. On considère le nombre de facettes de T appartenant à la surface du maillage.*

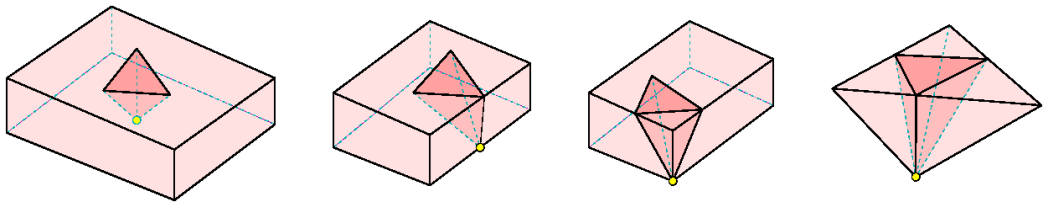


FIG. 3.5 – *Exemples de tétraèdres possédant une unique face appartenant à la surface du maillages.*

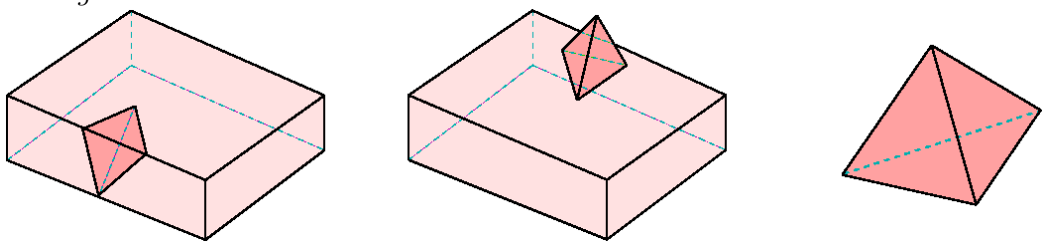


FIG. 3.6 – *Exemples de tétraèdres possédant 2, 3 ou 4 faces appartenant à la surface du maillage.*

- *Si le tétraèdre T n'a aucune de ses facettes appartenant à la surface du maillage alors T peut être retiré sans créer de singularité si, et seulement si, aucun de ses quatre sommets n'appartient à la surface du maillage. Notons que ce cas n'arrive jamais, les tétraèdres dont on souhaite le retrait possédant tous au moins une facette dans la surface du maillage (cf. § 4.4.6).*
- *Si le tétraèdre T a une et une seule de ses facettes appartenant à la surface du maillage, alors il peut être retiré si et seulement si le sommet opposé à cette facette n'appartient pas à la surface du maillage. Si le sommet opposé appartient à la surface du maillage il est encore possible de différentier quatre cas possibles suivant le nombre d'arêtes adjacentes à ce sommet qui appartiennent à la surface (cf. figure 3.5). Ce point sera discuté plus loin.*
  a. *aucune arête n'appartient à la surface,*
  b. *exactement une arête appartient à la surface,*
  c. *exactement deux arêtes appartiennent à la surface,*
  d. *les trois arêtes appartiennent à la surface.*
- *Si le tétraèdre T a exactement deux facettes appartenant à la surface du maillage, alors il peut être retiré si et seulement si l'arête opposée aux sommets communs à ces deux facettes n'appartient pas à la surface du maillage (cf. figure 3.6).*
- *Si le tétraèdre T a trois ou quatre facettes appartenant à la surface du maillage, il peut toujours être retiré (cf. figure 3.6).*

Preuve : On va prouver les affirmations précédentes cas par cas. Soit $\mathcal{T}$ un maillage variété et $\mathcal{T}'$ le maillage obtenu à partir de $\mathcal{T}$ après retrait d'un unique tétraèdre T. Il s'agit de montrer que la surface de ce maillage $\mathcal{T}'$ ne présente de singularité que

dans les cas référencés plus haut, et uniquement dans ces cas. On ne montrera que le premier cas, les autres démonstrations pouvant être effectuées de manière analogue.

Supposons que le tétraèdre T n'ait aucune de ses facettes dans la surface du maillage $\mathcal{T}$. La surface du maillage $\mathcal{T}'$ est donc composée d'une part des triangles du tétraèdre T et d'autre part des triangles de la surface du maillage $\mathcal{T}$, ces deux ensembles étant disjoints. Soit V un sommet de la surface du maillage $\mathcal{T}'$. Le voisinage de ce sommet est composé, d'une part, et dans le cas où ce sommet appartient à la surface du maillage $\mathcal{T}$, d'une composante connexe constituée du voisinage de ce sommet sur la surface de ce maillage, d'autre part, et dans le cas où ce sommet est l'un des sommets du tétraèdre T, d'une composante connexe constituée de trois des triangles de ce tétraèdre. On observe donc l'apparition d'une singularité située sur un sommet lorsque l'un des sommets du tétraèdre T appartient à la surface du maillage $\mathcal{T}$, et uniquement dans ce cas.
Supposons qu'aucun sommet de T n'appartienne à la surface de $\mathcal{T}$. Soit E l'une des arêtes de la surface du maillage $\mathcal{T}'$. Le voisinage de cette arête est composé, d'une part de deux triangles de la surface de $\mathcal{T}$, dans le cas où cette arête appartenait déjà à la surface de ce maillage, et d'autre part de deux des triangles du tétraèdre T dans le cas où cette arête est l'une des arêtes de ce tétraèdre. Aucun des sommets de T n'appartenant à la surface de $\mathcal{T}$, l'arête E ne peut appartenir à la fois au tétraèdre T et à la surface de $\mathcal{T}$. Son voisinage n'est donc constitué que de deux triangles. L'arête E ne présente donc pas de singularité.
On observe donc l'apparition d'une singularité sur le maillage $\mathcal{T}'$ lorsque l'un des sommets de T appartient à la surface du maillage $\mathcal{T}$, et uniquement dans ce cas.

◇

Savoir si une facette donnée du tétraèdre T appartient ou non à la surface du maillage consiste simplement à tester le caractère réel ou virtuel du tétraèdre voisin correspondant, ce qui est effectué par simple lecture du champ `tetraedre->neighbour[i]->empty` (cf. § 2.6.5). Déterminer l'appartenance d'un sommet donné à la surface du maillage revient à tester la nullité de son champ `vedge` (cf. § 2.6.2). Savoir si une arête donnée appartient ou non à la surface revient à tester son champ `vtriangle`.

La classification des différents cas peut donc être effectuée de manière efficace et très peu coûteuse en temps. On distingue quatre cas principaux, seuls les trois premiers posant problème :

- le tétraèdre peut être retiré sans créer de singularité,
- le tétraèdre T ne peut être retiré du maillage du fait d'une singularité située sur une arête,
- le tétraèdre T ne peut être retiré du maillage du fait d'une singularité située sur un sommet et l'une au moins des arêtes du tétraèdres adjacentes à ce sommet appartient à la surface du maillage,
- le tétraèdre T ne peut être retiré du maillage du fait d'une singularité située sur un sommet et aucune des arêtes du tétraèdres adjacentes à ce sommet n'appar-

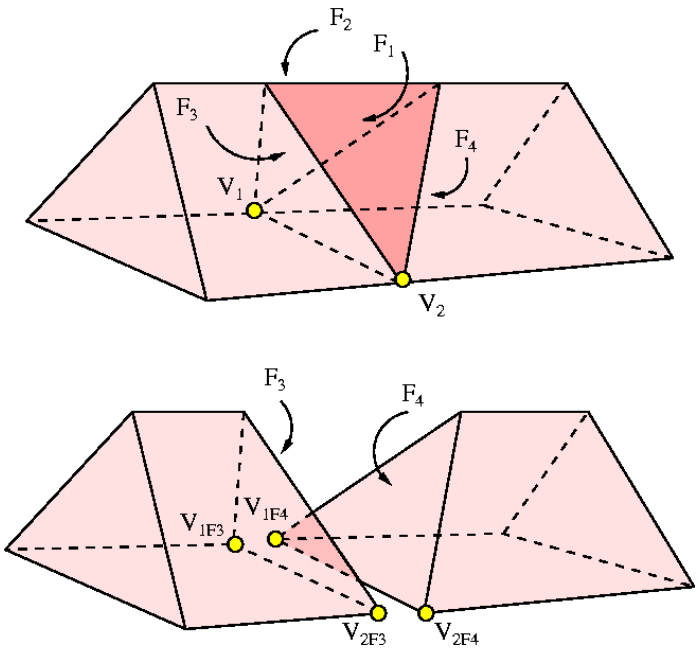


FIG. 3.7 – *Exemple de division d'une arête*

tient à la surface du maillage.

On agit alors en fonction du cas rencontré.

3.5 Singularité localisée sur une arête

Observation 3.5.1 *Si T est un tétraèdre dont exactement deux facettes appartiennent à la surface du maillage et dont le retrait cause une singularité, alors il est possible de supprimer cette singularité en scindant l'arête où elle est localisée.*

Preuve : Soit T le tétraèdre dont le retrait causerait une singularité. Soient F_1 et F_2 les facettes de ce tétraèdre appartenant à la surface du maillage. On a vu plus haut (cf. § 3.4) que cette singularité est située sur l'arête opposée à l'arête commune de ces deux faces. Soient E l'arête opposée à l'arête commune à ces deux faces, F_3 et F_4 les deux facettes de T adjacentes à E.

La facette F_1 et l'arête E ont un sommet commun et appartiennent tous deux à la surface du maillage. Le maillage étant conforme, F_1 et E sont donc concourantes dans la surface du maillage. De même, la facette F_2 et E sont concourantes dans la surface du maillage. Le tétraèdre T divise donc les tétraèdres, triangles et arêtes en contact avec l'arête E en deux composantes disjointes, l'une en contact avec F_3, l'autre en contact avec F_4. On peut donc scinder l'arête $E(V_1,V_2)$ en une arête $E_{F_3}(V_{1F_3},V_{2F_3})$ et une arête $E_{F_4}(V_{1F_4},V_{2F_4})$ Chaque apparition de E dans l'un des tétraèdres lui étant concourant sera remplacée par E_{F_3} si ce tétraèdre appartient à la composante connexe de F_3 et par E_{F_4} s'il appartient à la composante connexe de F_4. On procède de même

pour chaque apparition de V_1 et V_2 dans les arêtes, triangles et tétraèdres concourants à E.

Montrons que le maillage obtenu ne présente pas de singularité.

Notons tout d'abord que le maillage ne présente pas de singularité localisée sur des arêtes car la seule arête posant problème avant scission est E.

On peut considérer l'opération de scission comme une bijection entre l'ensemble des triangles de la surface du maillage avant scission et celui des triangles de la surface du maillage après scission. Les images de deux triangles adjacents sont alors deux triangles adjacents, sauf s'il s'agit de deux triangles adjacents à l'arête E. Réciproquement les antécédents de deux triangles adjacents sont deux triangles adjacents.

Si l'une des arêtes du maillage obtenu après scission était adjacente à quatre triangles, alors son antécédent le serait aussi. Or, seule l'arête E est en contact avec quatre triangles et les deux arêtes E_{F_3} et E_{F_4} obtenues par scission de E ne sont en contact chacune qu'avec deux triangles. Le maillage après scission ne présente donc pas de singularité localisée sur une arête.

Soit V un sommet de la surface du maillage avant la scission. Supposons que V ne soit pas l'un des sommets de E. Les images par la scission de deux triangles adjacents concourants à V étant adjacentes et ce voisinage étant connexe, l'image du voisinage de V est bien connexe. Supposons maintenant que V soit l'un des deux sommets de l'arête E. Le voisinage du sommet V avant le retrait du tétraèdre T est connexe par hypothèse. Le voisinage de ce sommet V après retrait l'est donc aussi car la seule différence avec le précédent est le remplacement du triangle F_1 (ou F_2) par les deux triangles adjacents F_3 et F_4. Ce voisinage a la particularité de posséder quatre triangles adjacents à l'arête E laquelle le divise donc en deux parties connexes distinctes. L'image de ces parties par la scission étant aussi connexe, les sommets V_{1F_3}, V_{1F_4}, V_{2F_3} et V_{2F_4} ne présentent donc pas de singularités. $\diamond$

3.6 Singularité localisée sur un sommet avec des arêtes adjacentes appartenant à la surface

La figure 3.8 montre des exemples de tels tétraèdres.

Observation 3.6.1 *Soit T un tétraèdre dont exactement une face F appartient à la surface du maillage et dont le retrait créerait une singularité située sur un sommet. Si l'une exactement des arêtes de T adjacentes à ce sommet appartient à la surface du maillage, alors il est possible de supprimer cette singularité en scindant celui des sommets de cette arête qui appartient aussi la face F.*

Observation 3.6.2 *Soit T un tétraèdre dont exactement une face appartient à la surface du maillage et dont le retrait créerait une singularité située sur un sommet.*

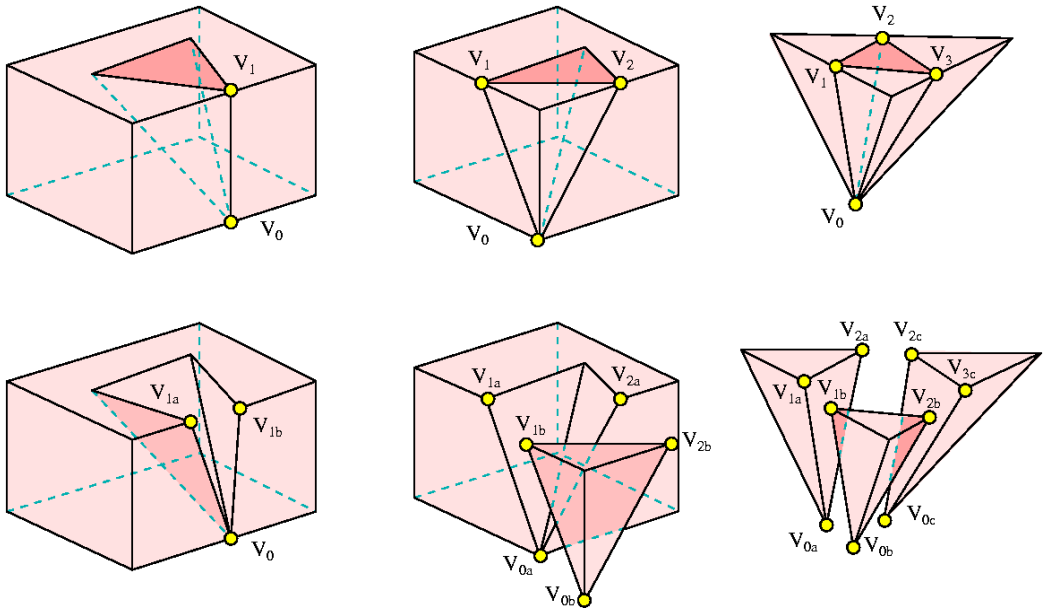


FIG. 3.8 – *Singularités topologiques localisées sur un sommet et pouvant être résolues à l'aide d'une simple scission d'arête ou de sommet.*

Si exactement deux des arêtes de T adjacentes à ce sommet appartiennent à la surface du maillage, alors il est possible de supprimer cette singularité en scindant les trois sommets de ses arêtes.

Observation 3.6.3 *Soit T un tétraèdre dont exactement une face appartient à la surface du maillage et dont le retrait créerait une singularité située sur un sommet. Si toutes les arêtes de T adjacentes à ce sommet appartiennent à la surface du maillage, alors il est possible de supprimer cette singularité en scindant les quatre sommets de T.*

Ces points se démontrent facilement d'une manière identique à celle utilisée précédemment dans le cas des singularités situées sur une arête (cf. § 3.5) et sont illustrés figure 3.8.

3.7 Singularité localisée sur un sommet sans aucune arête adjacente appartenant à la surface

3.7.1 Principe

Soit T le tétraèdre que l'on cherche à retirer. Supposons que le retrait de ce tétraèdre crée une singularité localisée sur un sommet et qu'aucune des trois arêtes de T adjacentes à ce sommet n'appartienne à la surface du maillage.

Le principe de la résolution est de déterminer un ensemble de tétraèdres $\mathcal{T}_{suppr}$, contenant le tétraèdre T, et dont le retrait du maillage ne crée pas de singularité. Il existe évidemment un grand nombre d'ensembles répondant à ces critères, le plus évident étant le maillage dans son ensemble. D'autres critères sont nécessaires pour permettre la détermination de $\mathcal{T}_{suppr}$. Le premier critère est la minimisation de la cardinalité de cet ensemble (ie. du nombre de tétraèdres à retirer). Il est en effet raisonnable de vouloir limiter au maximum les retraits de matière non désirés. Notons que ce critère aurait très bien pu être remplacé par une minimisation du volume total de l'ensemble $\mathcal{T}_{suppr}$. Nous verrons plus loin la raison de ce choix. Le deuxième critère est de ne considérer dans la recherche de $\mathcal{T}_{suppr}$ que des tétraèdres en contact avec T.

La raison de l'existence de ce critère est double. Premièrement, on cherche à limiter les effets visuels dus à un retrait de tétraèdres autres que celui désiré. Deuxièmement, subissant des contraintes de temps réel, on cherche à limiter le temps de calcul nécessaire à la détermination de l'ensemble $\mathcal{T}_{suppr}$. Malheureusement, cette limitation spatiale dans la recherche de $\mathcal{T}_{suppr}$ entraîne parfois l'absence de solution. La figure 3.9 montre un tel exemple dans le cas 2D. En 3D de tels cas sont possibles. Lorsque la recherche échoue, on décide de renvoyer l'ensemble vide ($\mathcal{T}_{suppr} = \emptyset$).

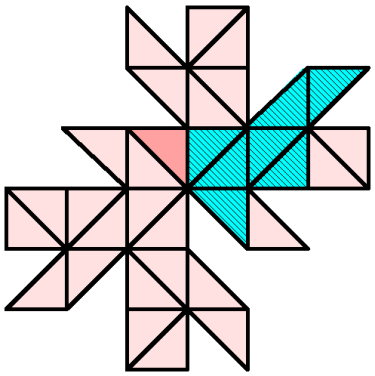


FIG. 3.9 – *Le retrait du triangle violet crée une singularité qui ne peut être supprimée que par le retrait de triangles ne lui étant pas adjacents*

3.7.2 Retrait d'un ensemble de tétraèdres

Pour pouvoir déterminer l'ensemble $\mathcal{T}_{suppr}$, il faut être capable de déterminer si un ensemble $\mathcal{T}$ donné peut ou non être retiré du maillage sans causer de singularité. Ce test est une opération sensiblement plus compliquée que la précédente. Elle revient à tester l'absence de singularité pour les sommets et les arêtes de la surface du maillage qui serait obtenu après le retrait des tétraèdres de $\mathcal{T}$. Ceci s'effectue de la façon suivante :

1. déterminer les ensemble $\mathcal{T}_v$, $\mathcal{T}_e$ composés des sommets et des arêtes de la nouvelle surface qui sont adjacents à l'ensemble $\mathcal{T}$,
2. vérifier que les arêtes de $\mathcal{T}_e$ ne sont adjacentes qu'à deux triangles de la surface du maillage qui serait obtenu par retrait de $\mathcal{T}$,
3. vérifier que le voisinage des sommets de $\mathcal{T}_v$ dans la surface du maillage qui serait obtenu par retrait de $\mathcal{T}$est bien connexe par les arêtes.

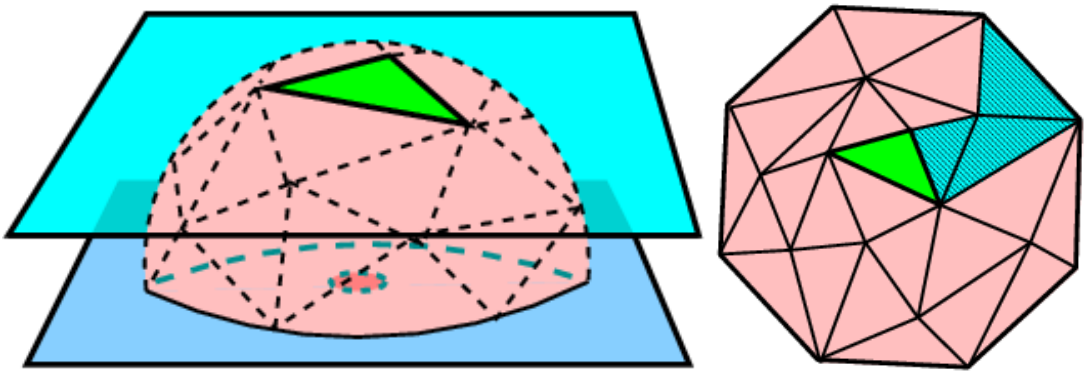


FIG. 3.10 – *Le voisinage du sommet problématique en représentation volumique et aplanie*

Remarquons qu'on se contente de tester les sommets et arêtes adjacents à l'ensemble $\mathcal{T}$, le voisinage des autres sommets et arêtes de la surface du maillage n'étant pas modifié par l'opération.

On débute le test par la détermination de l'ensemble $\mathcal{T}_t$ composé des triangles de la nouvelle surface du maillage. Cet ensemble peut être déterminé par un simple parcours des tétraèdres de $\mathcal{T}$. Si l'un des voisins de ces tétraèdres est réel mais n'appartient pas à l'ensemble $\mathcal{T}$, alors le triangle situé entre ces deux tétraèdres appartient à l'ensemble $\mathcal{T}_t$. Les ensembles $\mathcal{T}_v$ et $\mathcal{T}_e$ sont alors déterminés directement en effectuant l'union des sommets et des arêtes des triangles de l'ensemble $\mathcal{T}_t$

Les arêtes de $\mathcal{T}_e$ sont testées en parcourant leur voisinage et en vérifiant que ces voisinages ne sont pas coupés en deux par les tétraèdres de l'ensemble $\mathcal{T}$.

L'ensemble des triangles de la nouvelle surface concourants à un sommet donné V de $\mathcal{T}_v$ est l'union de l'ensemble des triangles de $\mathcal{T}_t$ concourants à V et de l'ensemble des triangles de l'ancienne surface concourants à V et n'étant pas adjacents à $\mathcal{T}$. Ce dernier ensemble est déterminé en parcourant le voisinage de l'éventuelle arête virtuelle concourante à V. On détermine alors le nombre de composantes connexes de cet ensemble.

Tester un ensemble de tétraèdres est toutefois une opération relativement coûteuse et il convient d'en limiter l'usage.

3.7.3 Détermination de l'ensemble retiré

Soit V le sommet posant problème. La figure 3.10 montre le voisinage de ce sommet V sous une représentation aplanie. Le retrait du tétraèdre T représenté en vert dans ces figures crée un trou dans le voisinage de V qui ne serait alors plus homéomorphe à une demi-sphère. L'idée est de supprimer ce trou en cherchant un chemin partant du tétraèdre à retirer et allant jusqu'à la surface, représentée par les arêtes extérieures de la représentation de la figure 3.10. La recherche de ce chemin est effectuée *en largeur d'abord* de façon à considérer en premier les chemins les plus courts. Elle est

de plus limitée à une profondeur (choisie arbitrairement) de 10 de façon à limiter le temps de recherche. Lorsqu'un chemin est trouvé, on teste l'ensemble de tétraèdres correspondant. En cas de succès, c'est cet ensemble qui sera renvoyé.

L'expérience nous a montré que l'échec de ce test est souvent dû au fait que l'ensemble choisi divise le voisinage de l'un des trois autres sommets de T en deux composantes connexes. On utilise donc une heuristique supplémentaire qui consiste à augmenter l'ensemble à retirer de l'une de ces deux composantes connexes (en commençant par la plus petite). Si l'un des deux ensembles $\mathcal{T}_{extended}$ ainsi obtenus peut être retiré sans problème du maillage, il sera utilisé si aucune meilleure solution n'est trouvée.

Si aucun ensemble n'est trouvé, on retourne l'ensemble vide.

3.8 Résultats

Afin d'estimer la validité de notre méthode, nous avons procédé à des opérations de découpe sur des maillages de différente nature. À chaque pas de la simulation, on a cherché à retirer les tétraèdres présents dans le voisinage immédiat de la pointe de l'outil virtuel. On a compté alors le nombre d'opérations de découpe ayant abouti ainsi que, pour chaque opération, le cardinal de l'ensemble effectivement retiré et la nature de l'éventuelle singularité détectée. Afin d'obtenir les chiffres les plus pertinents possibles, un même tétraèdre dont la tentative de retrait échoue à plusieurs reprises n'a été comptabilisé qu'une seule fois.

Pour cette étude on a utilisé les maillages suivants :

1. un maillage de cylindre,
2. un maillage de drap fin,
3. un maillage de foie d'environ 4000 tétraèdres obtenu directement par un mailleur volumique,
4. un maillage de foie d'environ 9000 tétraèdres obtenu à l'aide d'un mailleur volumique et raffiné localement à l'endroit de la découpe.
5. un maillage de foie d'environ 10000 tétraèdres raffiné "à la main" à l'aide d'une méthode de scission d'arêtes. Ce maillage est donc composé de tétraèdres de qualité sensiblement inférieure à ceux du maillage précédent.

Les résultats sont apparus comme étant relativement stables d'une expérimentation à l'autre et comme ne dépendant que faiblement de la nature du maillage, sauf dans des cas topologiques extrêmes tels que le drap fin et le cylindre. La tableau 3.1 montre, pour chacun des maillages testés, la proportion de tétraèdres pouvant être retirés sans problèmes et la proportion de ceux dont le retrait entraînerait la création d'une singularité topologique localisée sur un sommet ou sur une arête. Sauf pour le cas particulier du drap fin, la proportion de tétraèdres posant problème est d'environ 20%. Cependant, si ces tétraèdres n'étaient pas effectivement retirés du maillage, laur nombre augmenterait fortement, jusqu'à ce que la découpe devienne presque impos-

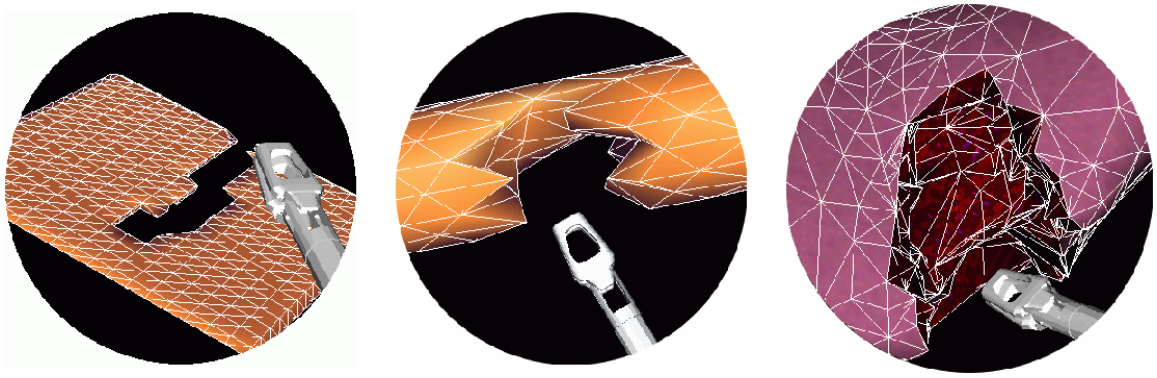


FIG. 3.11 – *Trois des cinq maillages utilisés pour l'évaluation de la méthode*

Maillage	nombre d'opérations	tétraèdres sans pb.	problème situé sur	
			un sommet	une arête
1	278	83 %	14 %	3 %
2	358	72 %	23 %	5 %
3	418	81 %	17 %	2 %
4	693	83 %	15 %	2 %
5	1881	83 %	15 %	2 %

TAB. 3.1 – *Répartition des singularités*

sible. Ceci montre l'importance de l'existence d'un algorithme permettant de réaliser malgré tout ces retraits.

Les singularités topologiques localisées sur des sommets sont de loin les plus communes car elles représentent plus de 90% des cas rencontrés. Le tableau 3.2 montre que la grande majorité de ces cas (80%) peut être résolue à l'aide de la méthode de scission que nous avons présenté précédement. L'algorithme que nous proposons est efficace car il résout plus de 99% des cas qui lui sont proposés. Notons que l'heuristique supplémentaire que nous proposons est elle aussi satisfaisant car elle permet de trouver une solution dans plus de la moitié des cas.

Pour finir, le tableau 3.3 montre la proportion de découpe réussies au cours de ces tests. Notre méthode a permit de retirer plus de 99,8% des tétraèdres rencontrées ce

Maillage	cas résolus			cas non résolus	card. moyen de l'ensemble retiré		
	scission	$\mathcal{T}_{suppr}$	heuristique		$\mathcal{T}_{suppr}$	heuristique	global
1	82 %	18 %	0 %	0	4,7	-	4,7
2	84 %	15 %	1 %	0	4,3	4,0	4,3
3	81 %	16 %	3 %	0	4,6	5,0	4,7
4	82 %	14 %	3 %	1	6,3	4,3	6,0
5	86 %	12 %	1 %	1	5,5	8,6	5,7

TAB. 3.2 – *Résolution du problème du sommet*

Mesh	General		Cas problématiques	
	opérations non résolues	card. moyen retiré	opérations non résolues	card. moyen retiré
1	0 %	1,1	0 %	1,4
2	0 %	1,1	1,0 %	1,3
3	0 %	1,1	2,5 %	1,4
4	0,15 %	1,1	3,4 %	1,5
5	0,15 %	1,1	1,8 %	1,4

TAB. 3.3 – *Pertinence de la solution*

qui est un très bon résultat, surtout si l'on tient compte du fait qu'au cours d'un pas de temps moyen du simulateur, le nombre de tétraèdres dont le retrait est demandé est généralement compris entre 4 et 6 (cf. § 4.4.6), ce qui diminue exponentiellement la probabilité d'un pas de temps sans aucun retrait de matière effectué.

Dans le cadre d'autres applications, et si on désire à tout prix effectuer une découpe donnée, on peut se rabattre sur une méthode à base de raffinement qui fournit toujours une solution, en gardant à l'esprit les problèmes de raffinements en cascade qu'elle peut générer.

3.9 Nécessité du raffinement

On a vu que l'une des contraintes principales que doit subir le simulateur pour le calcul des déformations est celle du temps réel. Les calculs doivent en effet être terminés suffisamment rapidement pour permettre le rafraîchissement visuel à une fréquence au moins égale à une vingtaine hertz. Pour augmenter les performances de la résolution, plusieurs solutions s'offrent à nous. Il est d'abord possible d'augmenter la puissance de calcul de la machine utilisée. On a en effet un lien direct entre la performance des calculs et la fréquence de fonctionnement des processeurs, et chaque génération de machine offre des gains sensibles dans ce domaine. Une autre possibilité consiste à améliorer les algorithmes de résolution, de façon à les rendre plus rapides ou à augmenter la vitesse de convergence. C'est ce qui est visé par les maillages hybrides [Cotin, 1997; Checoury, 2002]. Une dernière solution est de diminuer la taille du maillage, la durée des calculs étant généralement proportionnelle au nombre d'éléments de celui-ci.

Un maillage peu dense pose cependant des problèmes de réalisme. Les déformations calculées sont plus approximatives et le maillage est de plus en plus grossièrement facettisé et ressemble de moins en moins à l'organe modélisé.

Si ces problèmes peuvent être traités, par exemple en évitant d'utiliser des maillages exagérément grossiers ou en recouvrant la surface du maillage par une autre surface plus précise et liée à celle-ci (PN-Triangles [Vlachos *et al.*, 2001], Surfaces de subdivision [Debunne, 2000; Debunne *et al.*, 2001]), un problème demeure présent,

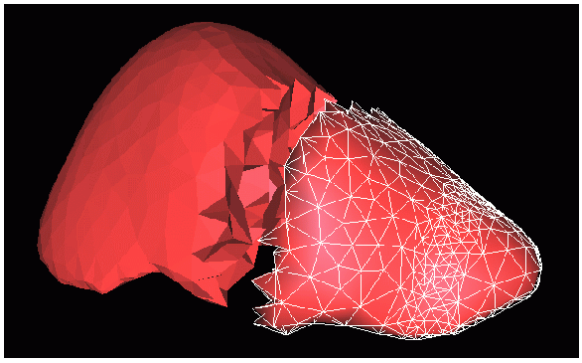


FIG. 3.12 – *Un exemple de foie partiellement maillé et préalablement raffiné. Seule la partie de droite est déformable.*

celui de la découpe. Telle qu'elle est conçue dans le simulateur, la découpe consiste en effet essentiellement à retirer certains tétraèdres du maillage. Si la taille de ces tétraèdres est trop importante, le résultat visuel devient désastreux.

Plusieurs solutions sont envisageables pour faire face au dilemme du choix entre rapidité de l'exécution et précision des déformations. La première consiste à diminuer le domaine du maillage. Moins de volume à mailler permet d'obtenir une densité de tétraèdres plus grande. Cette solution a été utilisée dans d'anciennes versions du simulateur : seul le tiers du foie correspondant aux régions concernées par l'opération est maillé, les deux autre tiers, supposés trop éloignés pour subir des déformations, restent fixe (voir figure 3.12) ou est animé à l'aide d'un modèle précalculé moins coûteux en temps de calcul [Cotin, 1997; Checoury, 2002]. Le défaut de cette approche est qu'il n'est pas possible d'interagir avec les parties du foie non maillées. Une autre idée est d'opérer sur un maillage dont seules les zones d'intérêt sont maillées de façon fine : dans notre cas il s'agit des zones ayant à subir la découpe. Cette idée est intéressante mais impose, avant de commencer l'opération, de déterminer avec précision les régions susceptibles d'être découpées et de s'y tenir. Faute de quoi les résultats visuels seront là aussi désastreux.

Les méthodes de multi-résolution proposent de faire cohabiter des maillages [Ganovelli *et al.*, 2000] ou des systèmes particulaires [Debunne, 2000] de différentes précisions. On peut passer localement d'un niveau de précision faible à un niveau de précision plus fin puis revenir à un niveau faible selon les besoins (précision de la découpe, visualisation, etc.). Ces méthodes peuvent être appliquées dans de nombreux domaines. On a par exemple relevé une application de la méthode de G.Debunne aux problèmes de détection de collisions entre objets de forme quelconque [Imagis, 2001, pp. 31–32]. Cette méthode ne permet cependant pas encore de découpe car la mise à jour des coefficients de rigidité lors de modifications topologiques telles que la découpe reste une opération délicate et non maîtrisée.

La méthode que nous avons choisie de mettre en œuvre consiste à modifier directement le maillage puis à le raffiner aux endroits et aux moments pertinents c'est-à-dire là où on cherche à découper. Pour pouvoir être intégrée dans le simulateur de chirur-

gie, cette procédure de remaillage doit, d'une part être rapide, et d'autre part ne pas générer de tétraèdres trop nombreux ou de trop mauvaise qualité.

3.10 Raffinement par subdivision

3.10.1 Généralités

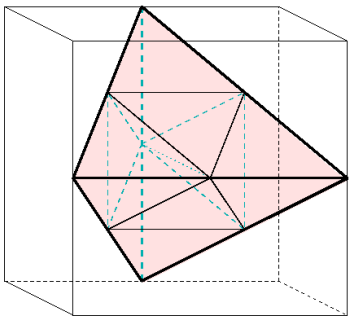


FIG. 3.13 – *Division du tétraèdre rhombique en huit tétraèdres semblables*

Les techniques de subdivision consistent à diviser les cellules existantes de façon a obtenir des cellules de taille plus petite. Elles ont l'avantage de proposer une solution simple et rapide au problème du raffinement local. Elles sont utilisées, par exemple, par les méthodes de résolution des élément finis utilisant des maillages adaptatifs [Löhner and Baum, 1992]. Ces techniques présentent l'intérêt d'être beaucoup plus rapides que d'autres méthodes de remaillage et de simplifier le passage des valeurs entre maillages de différentes résolutions.

Si les méthodes de raffinement de maillages triangulaires permettent facilement la division de triangles en triangles semblables, ce n'est pas le cas pour les maillages tétraédriques. Et une mauvaise gestion des opérations de subdivision peut dégrader très rapidement la qualité des éléments du maillage [Liu and Joe, 1996]. Des résultats ont cependant été obtenus pour certains types de tétraèdres. Par exemple, les tétraèdres des maillages dits CFK, (*Coexter-Freudenthal-Kuhn* [Coxeter, 1934]) qui résultent de la subdivision en six des cubes d'une grille régulière, peuvent être scindés en huit tétraèdres semblables [Persiano *et al.*, 1993]. Un même résultat est proposé pour le tétraèdre rhombique [Moore and Warren, 1990], qui est composé de quatre arêtes de longueur m et de deux de longueur $2m/\sqrt{3}$. Dans le cas général, des résultats théoriques ont montré qu'on peut borner la diminution de qualité des éléments raffinés pour des séries de bissections [Liu and Joe, 1995] ou des séries de subdivisions en huit tétraèdres [Liu and Joe, 1996].

Dans le cadre de raffinements locaux, il existe un certain nombre d'éléments dits *de transition* qui font l'interface entre les zones raffinées et non raffinées. Sauf dans le cas de certains maillages particuliers [Persiano *et al.*, 1993], ces éléments partiellement raffinés sont en effet d'une qualité relativement médiocre et un nouveau raffinement dégraderait encore sensiblement la qualité du maillage. S'il s'avère nécessaire de raffiner une zone contenant des éléments de transition, les méthodes classiques commencent généralement par "déraffiner" afin de revenir aux tétraèdres d'origine et de ne pas propager la chute de qualité. La simulation temps réel possède malheureusement un certain nombre de contraintes propres qui nous ont fait provisoirement renoncer à l'utilisation de telles méthodes. En particulier, les mécanismes permettant le déraffinement sont assez lourds et trop spécifiques. De plus, ce déraffinement peut

être rendu impossible par des modifications topologiques telles que la suppression de certains tétraèdres.

3.10.2 Qualité des tétraèdres

On s'inspire ici principalement des articles de Dompierre [Dompierre *et al.*, 1998], Liu [Liu and Joe, 1994] et du livre de Paul-Louis George [George and Borouchaki, 1997].

Définition 3.10.1 *On appelle tétraèdre* dégénéré, *un tétraèdre ayant un volume nul alors que l'une au moins de ses arêtes n'est pas de longueur nulle.*

Définition 3.10.2 *Soit Q une fonction de $(\mathbb{R}^3)^4$ dans* [0,1] *vérifiant les propriétés suivantes :*

- *Q est continue, et invariante par translation, rotation et homothétie,*
- *Q est maximum pour le tétraèdre régulier et minimum pour les tétraèdres dégénérés.*
- *Q ne présente pas de maxima ou de minima locaux autres que les minima et maxima globaux.*

On dit que Q est une fonction de qualité. *On appelle* qualité *d'un tétraèdre l'image d'un tétraèdre par une fonction de qualité. Il s'agit en fait d'une évaluation du degré de dégénérescence du tétraèdre.*

Observation 3.10.3 *Dans certaines applications, les fonctions utilisées ne discriminent pas certains types de tétraèdres dégénérés. C'est par exemple le cas des fonctions **Q4** et **Q5** présentées ci-après. Il ne s'agit donc plus de fonctions de qualité au sens de la définition 3.10.2. On pourra alors parler de* critère de qualité.

Définition 3.10.4 *On appelle tétraèdre* singulier *un tétraèdre possédant une "mauvaise" qualité. Il s'agit donc d'une notion subjective. On peut cependant différencier deux types de tétraèdres singuliers (cf. figure 3.14).*

> ***Type I :*** *l'une (au moins) des arêtes est très courte par rapport aux autres. On peut encore distinguer entre les tétraèdres ayant exactement une, deux et trois petite arêtes.*
>
> ***Type II :*** *les quatre sommets, sans être proche les uns des autres, sont presque coplanaires.*

Observation 3.10.5 *Lorsque les sommets forment un quadrilatère coplanaire, le tétraèdre est dit* en éclat *(cf. figure 3.14). L'appellation anglaise de* sliver *est aussi très utilisée. Ces tétraèdres sont relativement difficiles à détecter par des critères simples et sont un vrai problème pour certaines méthodes de maillage, par exemple les méthodes de maillage Delaunay (cf. § 3.11).*

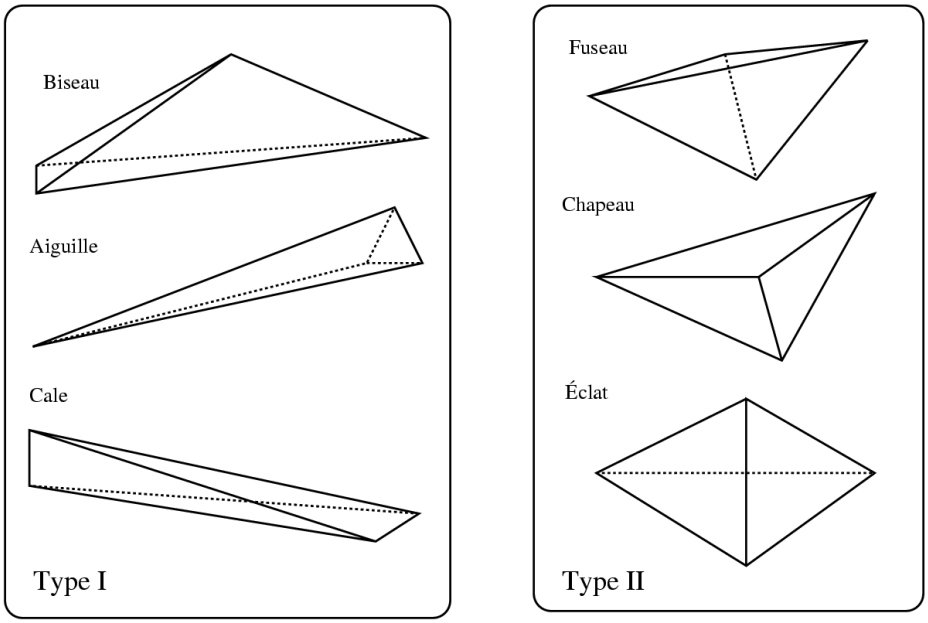


FIG. 3.14 – *Les différents types de tétraèdres singuliers*

Il existe un grand nombre de mesures de qualité différentes. L'intérêt de chacune des mesures est de simplifier les calculs dans telle ou telle application ou, lors d'opérations d'optimisation de maillage, de permettre de différencier les différents cas de tétraèdres singuliers afin de déterminer la meilleure méthode à utiliser pour la suite des opérations.

Q1 Le rapport ρ entre le rayon des sphères circonscrite et inscrite $\rho = \rho_{int}/\rho_{ext}$. On peut calculer facilement ces valeurs de la manière suivante :

$$\begin{aligned} \rho_{int} &= 3.v \Big/ \sum_{i=0}^{3} s_i \\ \rho_{ext} &= \frac{\sqrt{(a+b+c)(a+b-c)(a-b+c)(-a+b+c)}}{24.v} \\ \rho &= 3.\frac{\rho_{int}}{\rho_{ext}} \end{aligned}$$

où v est le volume du tétraèdre, s_i la surface de ses facettes et a, b et c le produit des longueurs des arêtes opposées.

Q2 Le rapport η entre moyennes géométrique et algébrique des valeurs propres de la matrice M transformant le tétraèdre en un tétraèdre régulier.

$$\eta = \frac{3.\sqrt[3]{\lambda_1\lambda_2\lambda_3}}{\lambda_1+\lambda_2+\lambda_3}$$

On peut simplifier ce calcul [Liu and Joe, 1995] :

$$\eta = \frac{12.\sqrt[3]{9.v^2}}{\sum_{0\le i\le j\le 3} l_{ij}^2}$$

Q3 L'angle solide θ_i d'un sommet donné correspond à la surface formée par la projection de ses trois sommets adjacents sur la sphère unité.

$$\begin{aligned} \theta_{min} &= \alpha . \min_{0\leq i\leq 3} \theta_i \\ \sin(\theta_i/2) &= 12.v\Big(\prod_{\substack{j,k\neq i \\ 0\leq j<k\leq 3}} \Big((l_{ij}+l_{ik})^2 - l_{jk}^2)\Big)\Big)^{-\frac{1}{2}} \end{aligned}$$

Où $\alpha^{-1} = 6.\arcsin(\sqrt{3}/3) - \pi$ est l'angle solide des sommets du tétraèdre régulier. Le calcul des θ_i se révèle coûteux à calculer car il passe par des fonctions trigonométriques inverses. On utilise plutôt les critères σ_{min} définis par :

$$\sigma_{min} = \beta . \min_{0\leq i\leq 3} \sin(\theta_i/2)$$

avec $\beta = \sin(\alpha^{-1}/2)$.

Q4 L'angle dièdre entre deux faces du tétraèdre est l'angle entre l'intersection de ces deux faces par un plan perpendiculaire à l'arête commune. Le critère correspondant ϕ_{min} est défini par :

$$\phi_{min} = \alpha \min_{0\leq i<j\leq 3} (\pi - \arccos(n_{ij1}.n_{ij2}))$$

où n_{ij1} et n_ij2 sont les normales des deux facettes adjacentes à l'arête ij et où $\alpha = \pi - \arccos(-1/3)$ est un coefficient de normalisation correspondant à l'angle dièdre entre deux facettes du tétraèdre isocèle.

Il ne s'agit pas d'une fonction de qualité au sens de la définition 3.10.2. En effet, ce critère ne discrimine pas les tétraèdres singuliers de type aiguille (voir figure 3.14).

Q5 Un autre critère couramment utilisé est le rapport r entre la longueur de l'arête la plus longue et celle de l'arête la plus courte.

$$r = \min_{0\leq i<j\leq 3} l_{ij} / \max_{0\leq i<j\leq 3} l_{ij}$$

Il ne s'agit pas non plus d'une fonction de qualité car il ne discrimine pas les tétraèdres singuliers de type éclat (voir figure 3.14).

Q6 Il existe encore bien d'autres critères. Comme que pour le premier critère Q1, ils sont généralement constitués du rapport entre deux grandeurs, l'une qui diminue avec le volume du tétraèdre, et l'autre non. Il peut s'agir par exemple du volume du tétraèdre, du rayon, de la surface ou du volume de la sphère inscrite, du plus petit des quatre angles solides d'une part, et d'autre part de la longueur de la plus grande arête, de la longueur moyenne ou de la somme des longueurs des arêtes, de la somme des surfaces des facettes, du rayon, de la surface ou du volume de la sphère circonscrite, etc. Les combinaisons sont infinies. Un exemple de tels critères est celui décrit par P-L George [George and Borouchaki, 1997] et utilisé entre autres par les mailleurs du projet GAMMA [GHS3D, 1999; GAMHIC3D, 2000].

$$\gamma = 2.\sqrt{6}\frac{\rho_{int}}{\max_{0\leq i<j\leq 0} l_{ij}}$$

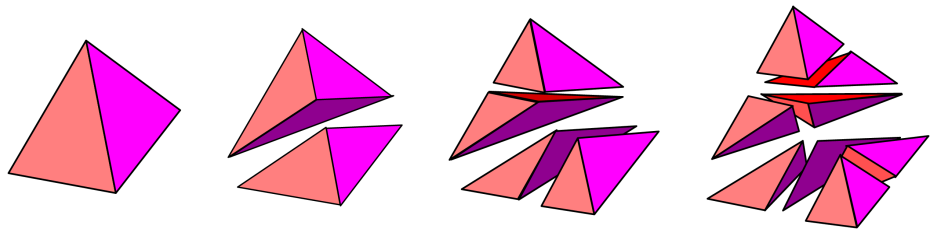


FIG. 3.15 – *Tétraèdre raffiné arête par arête*

Observation 3.10.6 *Liu [Liu and Joe, 1994] a montré qu'un grand nombre des fonctions de qualité présentées plus haut sont équivalentes, c'est-à-dire qu'il existe des constantes c_0, e_0, c_1 et e_1 telles que $c0.\mu^{e0} \leq \nu \leq c1.\mu^{e1}$. Le choix d'utiliser une fonction plutôt qu'une autre peut donc se baser uniquement sur des critères techniques de mise en œuvre ou de performance.*

3.10.3 Raffinement utilisé dans le simulateur

Les versions précédentes du simulateur utilisaient une méthode de raffinement qui procède par divisions successives d'arêtes [Picinbono, 2001]. Chacune des arêtes de la zone à raffiner est ainsi divisée une ou plusieurs fois, de façon à obtenir un maillage de la finesse désirée. La figure 3.15 montre un exemple de ce type de raffinement. La position exacte du point où se produit la division peut éventuellement être ajustée pour améliorer la qualité des tétraèdres résultants [Péquignot, 1999].

L'un des problèmes de cette méthode est sa lenteur. Pour obtenir les huit tétraèdres résultant de la découpe successive des quatre arêtes d'un tétraèdre de donné, il faut en effet préalablement construire, puis détruire, six tétraèdres (en plus de celui d'origine). Un grand nombre de calculs sont donc effectués inutilement.

Un autre problème est la faible qualité des tétraèdres générés, surtout si les arêtes sont divisées plusieurs fois. Le résultat du raffinement dépendant de l'ordre dans lequel sont traitées les arêtes, on peut cependant imaginer des heuristiques freinant la baisse de qualité. Par exemple scinder prioritairement les arêtes les plus longues, ou ne scinder une arête une deuxième fois que si toutes les arêtes des tétraèdres adjacents ont déjà été scindées une fois.

L'ordre dans lequel les arêtes sont considérées modifie grandement le résultat du raffinement (cf. figure 3.16). On utilise donc une heuristique qui nous interdit de raffiner une arête déjà raffinée si l'une des arêtes de l'un des tétraèdres adjacents ne l'a pas encore été. Dans le meilleur des cas, on ne pourra cependant pas faire mieux que le résultat théorique proposé par Liu [Liu and Joe, 1995] : si T_i^n est un tétraèdre obtenu après n bissections du tétraèdre T alors on a $\eta(T_i^n) \geq c.\eta(T)$ où c est une constante valant environ 0.1 et où η est la fonction de qualité correspondant au critère Q2 décrit précédemment.

Ces considérations nous ont conduits à améliorer la procédure de raffinement en

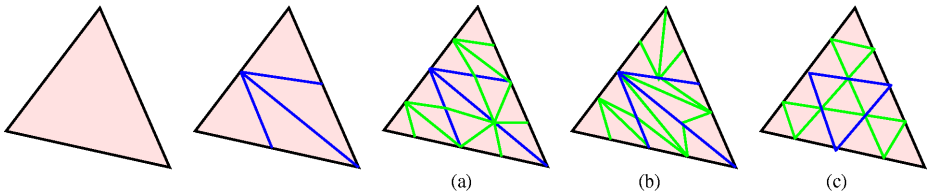


FIG. 3.16 – *Un triangle dont les arêtes sont raffinées, (a) dans l'ordre de leurs création (b) au hasard et (c) simultanément*

scindant les arêtes de manière simultanée et non plus une à une. Le premier intérêt est l'accélération de la procédure du fait de l'absence de création de tétraèdres provisoires. Le second est la nette amélioration de la qualité des tétraèdres. Liu démontre en effet [Liu and Joe, 1996] que la qualité des tétraèdres T_i résultant d'une scission en huit d'un tétraèdre T est bornée par : $\eta(T_i) \leq 0{,}5.\eta(T)$.

Ces résultats ne sont cependant pas applicables aux éléments de transition qui ne sont que partiellement raffinés. Cela est particulièrement sensible compte-tenu du fait que, comme on l'a dit plus haut, ces éléments de transition ne sont pas marqués comme tels et ne peuvent donc pas faire l'objet d'un traitement spécifique lors de futurs raffinements. Pour estimer la qualité des tétraèdres de transition, on s'est orienté vers la fonction de qualité r, rapport entre la longueur de l'arête la plus courte et celle de l'arête la plus longue, qui correspond au critère **Q5** décrit précédemment. Cette fonction a le défaut de ne pas détecter les dégénérescences de type II mais elle est extrêmement aisée à calculer. Il est de plus très commode de contrôler sa valeur par les mécanismes de scission d'arêtes que nous utilisons.

La première étape du processus consiste à déterminer l'ensemble $\mathcal{E}$ des arêtes à raffiner. Il s'agit de toutes les arêtes de la zone à raffiner dont la longueur est supérieure à une longueur caractéristique l_{max} qui est la longueur maximale que l'on désire avoir dans le maillage raffiné. Dans le cadre des opérations de découpe du simulateur, la zone à raffiner est constituée de l'ensemble $\mathcal{T}$ des tétraèdres situés a proximité de la pointe de l'outil que l'on détermine de la manière suivante. Dans un premier temps, une opération de détection de collision entre l'outil et le maillage (cf. § 4.4.1) permet de déterminer l'ensemble des triangles de la surface situés à proximité de l'outil. L'ensemble $\mathcal{T}$ est constitué de l'ensemble des tétraèdres concourants à l'un des sommets de ces triangles et situés à une distance de la pointe de l'outil inférieure à deux fois le rayon de celui-ci. La détermination de cet ensemble utilise les itérateurs présentés dans la partie 2.6.2 et est donc facile à réaliser. L'ensemble $\mathcal{E}$ regroupe alors les arêtes des tétraèdres de $\mathcal{T}$ dont la longueur est plus grande que l_{max}. La valeur de l_{max} est réglable mais on la prend généralement de l'ordre du diamètre de l'outil.

Pour éviter une diminution trop rapide de la qualité des tétraèdres situés en bordure de la zone à raffiner, on récupère l'ensemble $\mathcal{V}(\mathcal{E})$ constitué des tétraèdres adjacents à au moins l'une des arêtes de $\mathcal{E}$. On rajoute alors à $\mathcal{E}$ toutes les arêtes de ces tétraèdres dont la longueur est plus grande que 2 fois la longueur de son arête la plus courte Ce dernier point est répété autant de fois que nécessaire. Cette opération cor-

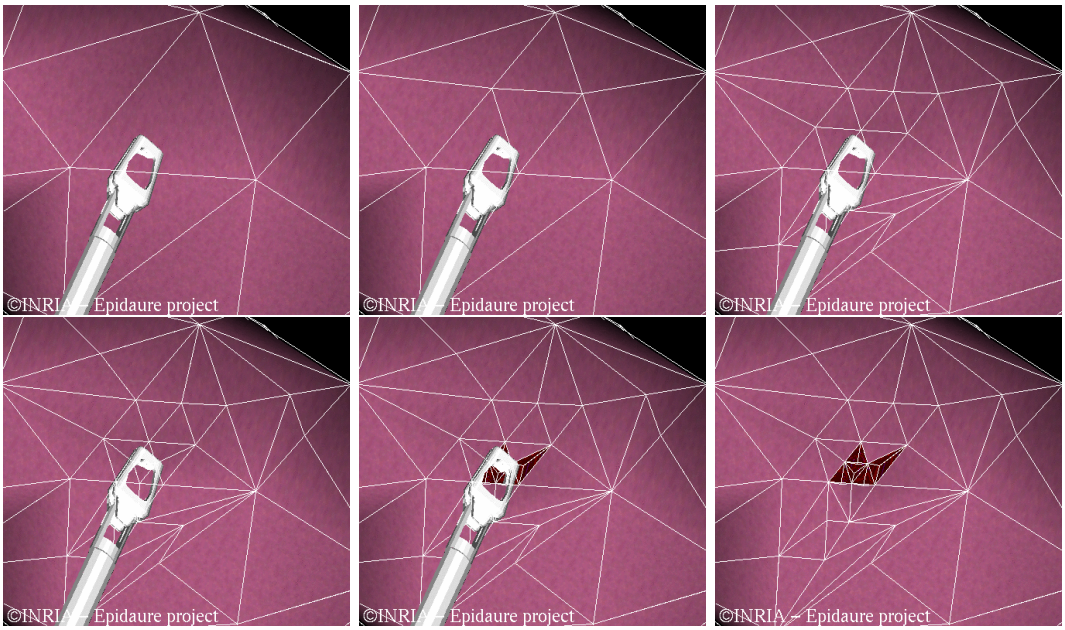


FIG. 3.17 – *Les différentes étapes de l'opération de découpe : le maillage à proximité de la pointe de l'outil est raffiné une ou plusieurs fois, en fonction de la taille des tétraèdres, préalablement à la découpe proprement dite.*

respond en fait à peu près à l'utilisation du critère **Q5** présenté précédemment. Elle permet de propager le raffinement à un certain nombre de tétraèdres pour lesquels la valeur de ce critère **Q5** serait sinon inférieur à 0,25.

Chacun des tétraèdres de $\mathcal{T}_\mathcal{E}$ est alors divisé en fonction du nombre d'arêtes de $\mathcal{E}$ qui lui sont adjacentes. Les opérations de division s'effectuent tétraèdre par tétraèdre. La description exacte de la manière dont sont divisés les tétraèdres est précisée figure 3.18. Pour garantir la conformité du maillage dans le cas où seules deux arêtes d'une face donnée sont scindées, on a dû introduire un ordre sur les sommets. Par raison de simplicité, on a choisi de comparer la valeur de leurs pointeurs, mais tout autre ordre total serait aussi valable. L'opération est répétée autant de fois que nécessaire jusqu'à ce qu'il ne reste plus d'arête de longueur supérieure à l_{max} dans la zone d'intérêt.

3.11 Remaillage de type Delaunay

Les méthodes de raffinement par subdivision sont intéressantes du fait de la simplicité de leur mise en œuvre. Elles sont cependant peu performantes quant à la qualité des tétraèdres générés. Une méthode évoquée récemment par Nienhuys [Nienhuys and van der Stappen, 2002] propose de remailler la zone d'intérêt à l'aide d'une méthode de type Delaunay. Un maillage tétraédrique est dit Delaunay [Delaunay, 1934; George and Borouchaki, 1997] lorsque la sphère circonscrite de chaque tétraèdre du maillage ne contient aucun sommet autre que ceux de ce même tétraèdre (voir figure 3.19). Les maillages de Delaunay possèdent un grand nombre de qualités, entre

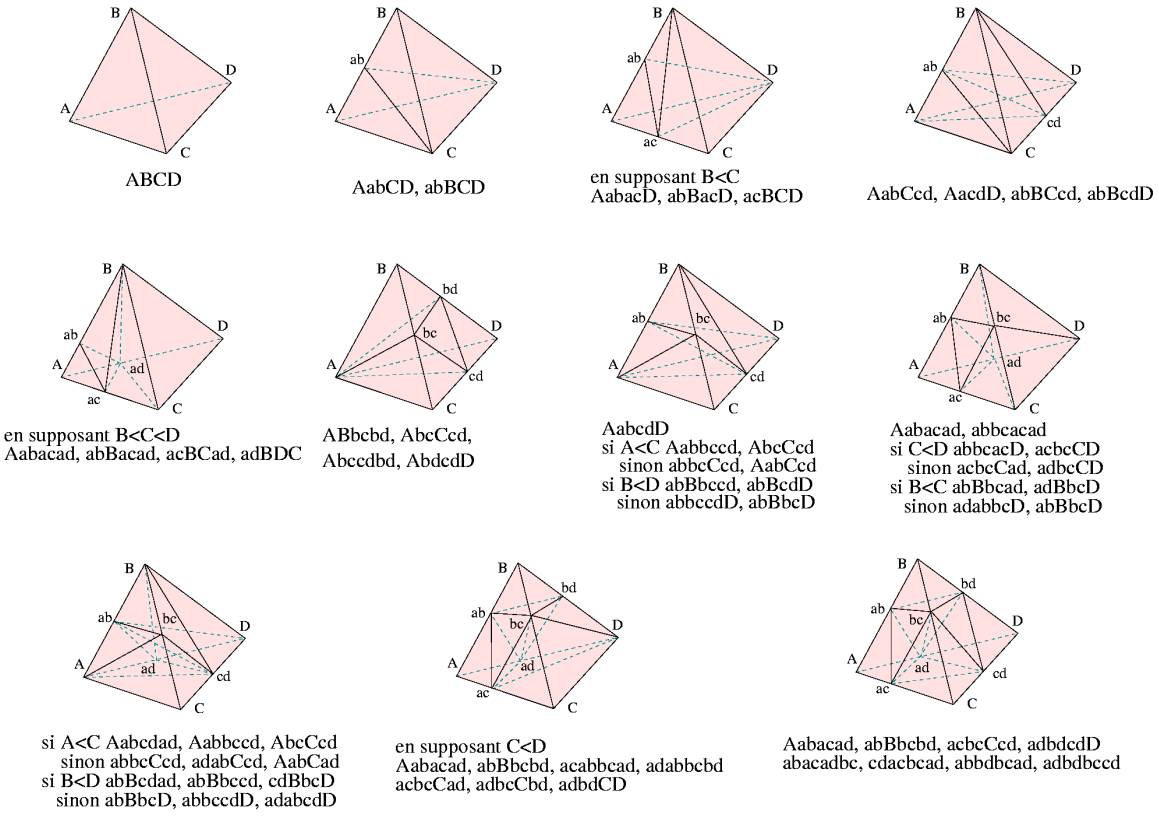


FIG. 3.18 – *Les différents cas possibles lors de la scission des arêtes d'un tétraèdre. On suppose qu'il existe une relation d'ordre entre les sommets de façon à préserver la conformité du maillage. Cette relation d'ordre est utile lorsque seules deux des trois arêtes d'un triangle donné sont raffinées.*

autres leur unicité étant donné un ensemble de sommets donnés (sauf cas particuliers, lorsque plusieurs sommets sont cocycliques) et leur facilité de construction. De plus, si les sommets sont bien répartis, la qualité des tétraèdres générés est relativement acceptable, même s'il n'y a pas de garantie théorique à cela. En particulier, les tétraèdres dégénérés de type éclat sont tout à fait admissibles par les triangulations de Delaunay, ce qui fait que les mailleurs traditionnels sont rarement Delaunay, ou au moins disposent d'autres critères permettant l'amélioration du maillage.

L'idée évoquée par Nienhuys consiste donc à rajouter ou même à déplacer un certain nombre de sommets et de reconstruire localement le maillage à l'aide d'une méthode de Delaunay. Pour notre cas, l'intérêt serait d'une part de permettre une vraie maîtrise de la précision désirée, fonction du nombre de sommets rajoutés, et cela sans avoir à procéder à plusieurs maillages successifs, et d'autre part de supprimer les problèmes liés à la qualité des éléments de transition. Cette méthode pose cependant certains problèmes qui ont jusqu'à présent empêché sa mise en œuvre. Le remaillage que l'on souhaite effectuer doit en effet préserver la surface du maillage ainsi que les éventuelles sous-structures qui le composent (tumeur, segment de Couineau, etc.) : on parle alors de triangulation de Delaunay *contrainte*, et cette exigence

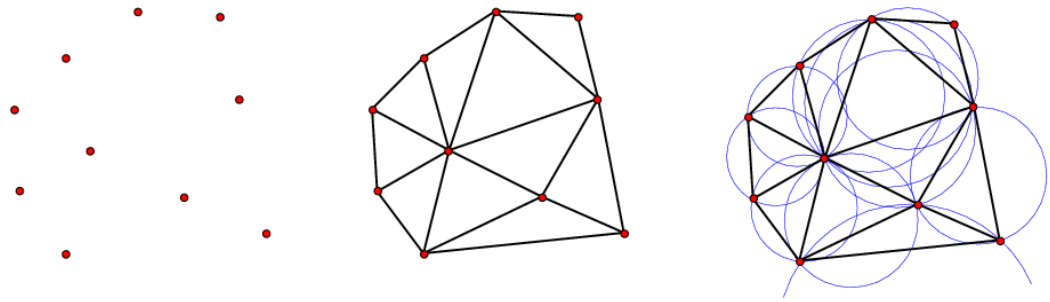


FIG. 3.19 – *À gauche, un nuage de points de $\mathbb{R}^2$. Au centre, sa triangulation de Delaunay. À droite, on a représenté l'ensemble des cercles circonscrits.*

peut parfois empêcher la convergence des algorithmes de remaillage, surtout dans le cas de contraintes présentant des angles très faibles [Shewchuk, 2000]. C. Paloc [Paloc *et al.*, 2002] propose cependant une méthode de raffinement dynamique de maillage volumique utilisant les propriétés de Delaunay.

À notre sens, c'est cependant vers ces méthodes de remaillage local (mais pas forcément Delaunay) qu'il faut se diriger. Les sommets et les tétraèdres du maillage ne sont en effet que des supports à la description de la géométrie de l'objet et ne devraient pas être des contraintes empêchant l'obtention d'un maillage raffiné de qualité.

3.12 Conclusion

Nous avons défini la notion de maillage volumique variété et rappelé l'importance de cette notion pour la résolution de nombreux problèmes liés à la simulation de chirurgie : visualisation, retour d'effort, raffinement, etc. Nous avons présenté une méthode de découpe de maillage par retrait de tétraèdres préservant la variété du maillage et utilisable dans le cadre d'un simulateur de chirurgie. Nous avons dû préalablement répertorier et classifier les différentes singularités qui peuvent apparaître dans un maillage suite au retrait d'un unique tétraèdre. Dans certains cas, la méthode que nous proposons ne fournit pas de solution : la découpe est alors impossible. Nous avons montré que ces cas sont cependant suffisamment peu nombreux pour ne pas être gênants dans le cadre de notre application.

Pour être visuellement acceptable, cette méthode doit cependant opérer sur des tétraèdres de taille suffisamment réduite. Nous proposons donc une méthode de raffinement dynamique du maillage qui, limitant la perte de qualité des tétraèdres permet de débuter la simulation avec des maillages de faible résolution. Cette méthode fonctionne par subdivision et augmente donc rapidement le nombre de tétraèdres dans les zones raffinées. Enfin, il serait intéressant de développer des méthodes plus progressives telles que la méthode de type Delaunay évoquée précédemment.

Chapitre 4

Interface avec l'environnement virtuel

Sommaire

4.1 Introduction

La possibilité d'interagir avec le modèle mécanique simulé permet d'augmenter considérablement le réalisme.

On peut distinguer deux grands types de périphériques utilisés en simulation de chirurgie. D'une part, on a les périphériques dédiés à un type d'utilisation bien défini, par exemple la chirurgie laparoscopique [Lamy and Chaillou, 1999; Baur *et al.*, 1998], ou encore plus spécifique, la chirurgie du genou [Hollands and Trowbridge, 1996] ou la photocoagulation laser [Meseure *et al.*, 1995]. On peut aussi rajouter à cette liste les parties maîtres de certains systèmes de télé-chirurgie [Woo *et al.*, 1998]. L'intérêt de ces périphériques est qu'ils représentent de manière très réaliste les outils utilisés lors des opérations réelles. D'autre part, on a les périphériques d'usage plus général qui peuvent cependant être utilisés dans certaines applications de simulation de chirurgie. On peut citer le gant à retour d'effort, que l'on trouve par exemple en simulation de palpation [Michael *et al.*, 1997] ou encore en simulation de télé-chirurgie [Bar-Cohen *et al.*, 2001]. Un autre périphérique générique très populaire est le PhantoM, développé

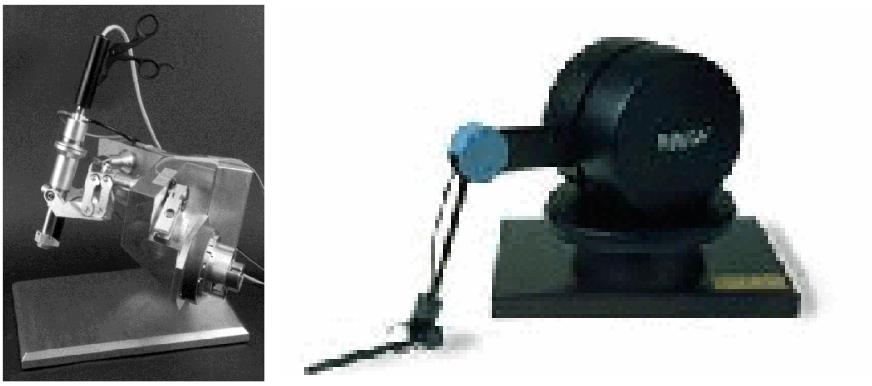


FIG. 4.1 – *Gauche : le PantaScope, développé à l'EPFL (source [Baur* et al.*, 1998]). Droite : l'un des modèles de PhantoM de Sensable (source* `www.sensable.com`*).*

par la société Sensable Technologies [Massie and Salisury, 1994]. Le PhantoM possède six degrés de liberté, à savoir la position et l'orientation. Dans le cas de son modèle le plus courant, les trois degrés de position sont actifs, c'est-à-dire qu'on peut y exercer des forces. L'absence de retour d'effort sur les trois degrés d'orientation fait qu'il est plus adapté à la simulation d'outils ponctuels tels que la pointe d'un stylet. Il existe cependant certains modèles, encore peu répandus, proposant six degrés de liberté pour le retour d'effort [Chen, 1999]. Le nombre de degrés de liberté peut aussi être diminué de façon à simuler un instrument de laparoscopie [Tendick *et al.*, 2000]. Une interface permettant d'utiliser le PhantoM à l'intérieur du logiciel `yav++` a été développé [Tonet, 2002].

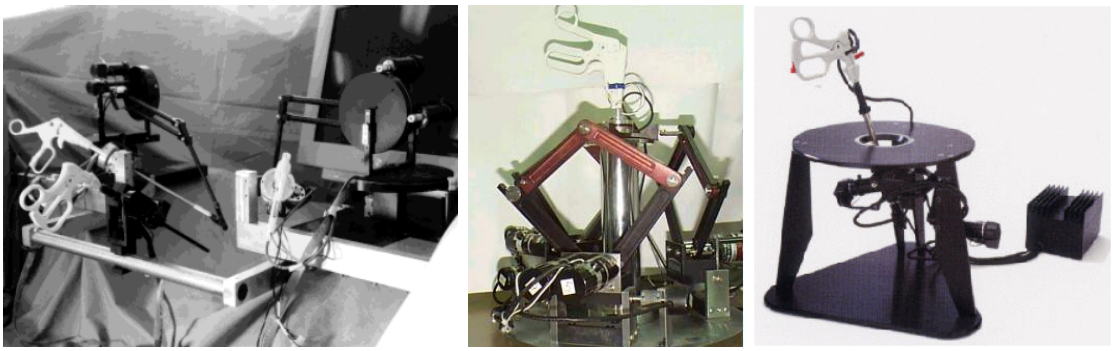


FIG. 4.2 – *Gauche : système à retour d'effort du projet Vesta de Berkeley. Le PhantoM de Sensable est adapté de façon à faire passer l'outil par un point fixe (source [Tendick* et al.*, 2000]). Centre : le bras maître à 6 degrés de liberté du système de télé-chirurgie du Korea Advanced Institute of Science and Technology (KAIST) (source* `robot.kaist.ac.kr`*). Droite : le LIE de Immersion Corp. (source* `www.immersion.com`*).*

4.2 Le Laparoscopique Impulse Engine

4.2.1 Description Générale

Le Laparoscopique Impulse Engine (LIE) est un périphérique développé par la société Immersion Corp. Il a été conçu pour simuler les outils couramment utilisés en chirurgie laparoscopique : la poignée est, par exemple, la réplique exacte de celle d'un véritable instrument.

Le LIE est constitué d'un axe maintenu au bâti par un système composé de cinq liaisons axiales simulant une liaison rotule, et d'une liaison glissière. Cette articulation, décrite figure 4.4, contraint l'axe à passer par un *point fixe*, de la même façon que l'axe de l'outil passe par le point d'insertion du trocart. Le débattement est cependant réduit : en opération, le chirurgien dispose d'un champ de travail égal à une portion de sphère de rayon 30 cm et d'angle solide proche de 180^o alors que le LIE est limité à un rayon de 8 cm et à un angle solide de 60^o. La liaison glissière (Z) et deux des liaisons axiales (X et Y) sont reliées à des petits moteurs de 20 W, de type maxon RE025 [Maxon Motor ag, 2002], permettant l'envoi de forces. Le contrôle de la position s'effectue à l'aide de trois encodeurs optiques US Digital de type HEDS-5500 [Digital Corporation, 2002]. La rotation autour de l'axe Z est déterminée à l'aide d'un encodeur Bourns de type ENT1D [Bourns, 2002]. Moteurs et encodeurs sont reliés à l'ordinateur par le biais d'une carte PCI dédiée.

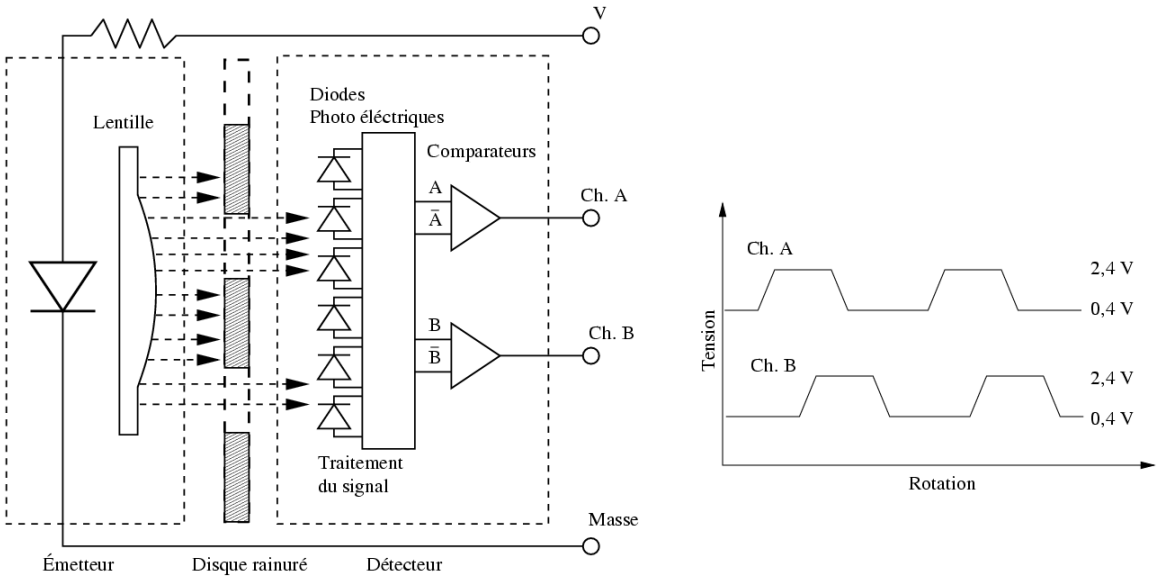


FIG. 4.3 – *Schéma de fonctionnement d'un encodeur optique simple*

4.2.2 Détermination de la position

Description mécanique

Le principe de fonctionnement des encodeurs optiques est relativement simple et est décrit figure 4.3. Une diode électroluminescente est utilisée comme source de lumière dirigée sur un ensemble de diodes photoélectriques. L'axe dont on veut contrôler la rotation est relié solidairement à un disque rainuré qui cause l'occlusion périodique du faisceau lumineux. Le circuit de traitement du signal génère alors deux signaux carrés A et B en quadrature. Si l'axe tourne dans le sens trigonométrique (en regardant vers le moteur), la sortie A est en avance sur la sortie B. S'il tourne dans le sens contraire, c'est la sortie B qui est en avance. Il est ainsi possible de déterminer le sens de rotation de l'axe. À chaque fois qu'une rainure passe devant la lentille, une impulsion + ou − est envoyée à un additionneur situé sur la carte PCI. Une simple lecture de la valeur de celui-ci permet donc de connaître la rotation exacte de l'axe correspondant par rapport à une position de référence appelée *position de repos*. Par commodité, on définit cette position comme celle où l'outil est centré sur l'axe Z le plus haut possible et où la poignée est alignée le long de l'axe X.

Soient α, β et δ les angles de rotation des axes Y X et Z par rapport à la position de repos. Soit l l'enfoncement de la poignée par rapport à cette même position. Les disques des encodeurs étant rainurés de façon régulière, on a :

$$\alpha = \frac{\Delta\alpha}{\Delta IO_0} * IO_0 \tag{4.1}$$

$$\beta = \frac{\Delta\beta}{\Delta IO_1} * IO_1 \tag{4.2}$$

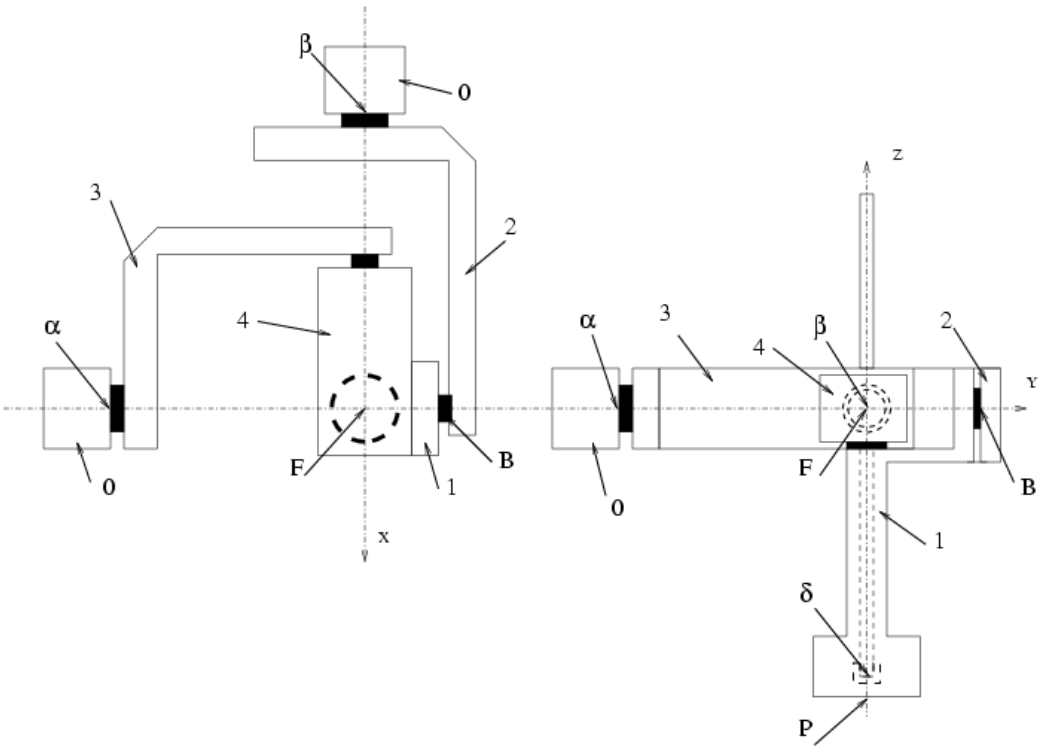


FIG. 4.4 – *Schéma mécanique du LIE*

$$l = \frac{\Delta l}{\Delta IO_2} * IO_2 \tag{4.3}$$

$$\delta = \frac{\Delta \delta}{\Delta IO_3} * IO_3 \tag{4.4}$$

où IO_0, IO_1, IO_2 et IO_3 représentent la valeur des quatre additionneurs, $\Delta\alpha$, $\Delta\beta$, Δl et $\Delta\delta$ sont les débattements maximum des axes correspondants et ΔIO_0, ΔIO_1, ΔIO_2 et ΔIO_3 l'écart de valeur des additionneurs correspondant à ces débattements maximum. Pour ces derniers, on trouve les valeurs :

$$\begin{array}{llll} \Delta\alpha & = 60^o & = \frac{1}{3}.\pi, & \Delta IO_0 = 5000 \\ \Delta\beta & = 60^o & = \frac{1}{3}.\pi, & \Delta IO_1 = 5000 \\ \Delta l & = 8cm, & & \Delta IO_2 = 5800 \\ \Delta\delta & = 360^o & = 2.\pi, & \Delta IO_3 = 1024 \end{array}$$

Notons qu'il est possible d'effectuer plus d'un tour à l'outil autour de l'axe Z. On a donc pris arbitrairement des valeurs de $\Delta\delta$ et de IO_3 correspondant à un tour complet autour de cet axe.

Comme évoqué précédemment, le débattement des axes α, β et l est très inférieur aux valeurs réelles accessibles lors d'une véritable opération par cœlioscopie.

Détermination de la position de l'outil

On cherche à déterminer la position du point P situé au bout de l'axe Z de l'outil. La figure 4.4 représente le schéma mécanique du LIE. On peut distinguer quatre parties mobiles que l'on nommera $S1$, $S2$, $S3$ et $S4$. Soit $\mathcal{R}_{S0} = (\overrightarrow{X}, \overrightarrow{Y}, \overrightarrow{Z})$ le référentiel lié au bâti de l'outil. Soient $\mathcal{R}_{S1}$, $\mathcal{R}_{S2}$, $\mathcal{R}_{S3}$ et $\mathcal{R}_{S4}$ les quatre référentiels liés aux parties mobiles du système. On suppose que pour la position de repos, ces quatre référentiels sont égaux à $\mathcal{R}_{S0}$.

Observation 4.2.1 *Si F est le point d'intersection des axes du dispositif dans la position de repos, alors on a :*

$$P = F - (l + l_0).\overrightarrow{Z}_{S1} \tag{4.5}$$

où l_0 est la distance séparant ce point F du point P pour la position de repos.

Preuve : En effet, on peut remarquer que tous les axes du dispositif sont alignés sur ce même point F qui est donc immobile pour la totalité de ces référentiels. Le point P étant solidaire du solide $S1$, l'égalité précédente, vraie pour la position de repos, reste valable à tout instant. $\diamond$

Cherchons maintenant à déterminer le référentiel $\mathcal{R}_{S1}$. La pièce $S2$ et le bâti étant reliés par une liaison axiale d'axe $\overrightarrow{X}$, et dont l'angle de rotation est égal à β, on a :

$$\begin{aligned} \overrightarrow{X}_{S2} &= \overrightarrow{X} \\ \overrightarrow{Y}_{S2} &= \overrightarrow{Y}.\cos(\beta) + \overrightarrow{Z}.\sin(\beta) \end{aligned}$$

Les pièces $S1$ et $S2$ étant liées entre elles par une liaison axiale orientée selon les $\overrightarrow{Y}$, les deux axes $\overrightarrow{Y}_{S1}$ et $\overrightarrow{Y}_{S2}$ sont identiques. On a donc :

$$\overrightarrow{Y}_{S1} = \overrightarrow{Y}_{S2} = \overrightarrow{Y}.\cos(\beta) + \overrightarrow{Z}.\sin(\beta) \tag{4.6}$$

La pièce $S3$ et le bâti étant reliés par une liaison axiale d'axe $\overrightarrow{Y}$, et dont l'angle de rotation vaut α, on a :

$$\begin{aligned} \overrightarrow{Y}_{S3} &= \overrightarrow{Y} \\ \overrightarrow{X}_{S3} &= \overrightarrow{X}.\cos(\alpha) - \overrightarrow{Z}.\sin(\alpha) \end{aligned}$$

Les pièces $S3$ et $S4$ étant reliées par une liaison axiale orientée selon les $\overrightarrow{X}$, les deux axes $\overrightarrow{X}_{S4}$ et $\overrightarrow{X}_{S3}$ sont identiques. On a donc :

$$\overrightarrow{X}_{S4} = \overrightarrow{X}_{S3} = \overrightarrow{X}.\cos(\alpha) - \overrightarrow{Z}.\sin(\alpha)$$

Les pièces $S4$ et $S1$ étant reliées par une liaison axiale orientée selon les $\overrightarrow{Z}$, les deux axes $\overrightarrow{Z}_{S4}$ et $\overrightarrow{Z}_{S1}$ sont identiques. L'axe $\overrightarrow{Z}_{S1}$ est donc orthogonal, d'une part à $\overrightarrow{X}_{S4}$,

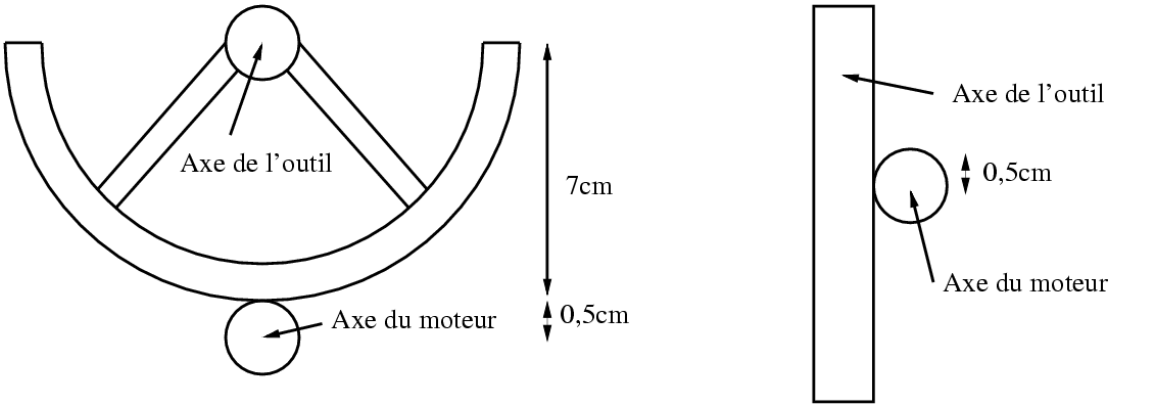


FIG. 4.5 – *Schéma des réducteurs du LIE. Gauche : réducteur des axes X et Y. Droite : réducteur de l'axe Z*

et d'autre part à $\overrightarrow{Y}_{S1}$. On peut donc le calculer :

$$\begin{aligned}
\overrightarrow{Z}_{S1} &= \lambda.\left(\overrightarrow{X}_{S4} \wedge \overrightarrow{Y}_{S1}\right) \\
&= \lambda.\left(\overrightarrow{X}.\cos(\alpha) - \overrightarrow{Z}.\sin(\alpha)\right) \wedge \left(\overrightarrow{Y}.\cos(\beta) + \overrightarrow{Z}.\sin(\beta)\right) \\
&= \lambda.\left(\overrightarrow{X}.\cos(\beta).\sin(\alpha) - \overrightarrow{Y}.\cos(\alpha).\sin(\beta) + \overrightarrow{Z}.\cos(\alpha).\cos(\beta)\right) \qquad (4.7) \\
& \qquad (4.8)
\end{aligned}$$

Ce vecteur étant unitaire, on a pour valeur de λ :

$$\lambda = \left\|\overrightarrow{X}_{S4} \wedge \overrightarrow{Y}_{S1}\right\| = \cos(\alpha)^2 + \cos(\beta)^2 - \cos(\alpha)^2.\cos(\beta)^2$$

Détermination de l'orientation de l'outil

Soit $\overrightarrow{A}$ l'axe d'ouverture de la poignée. Pour la position de repos, cet axe est égal à $\overrightarrow{Y}$. On a donc :

$$\overrightarrow{A} = \overrightarrow{X}_{S1}.\sin(\delta) + \overrightarrow{Y}_{S1}.\cos(\delta) \qquad (4.9)$$

Il ne reste donc plus qu'à calculer $\overrightarrow{X}_{S1}$:

$$\begin{aligned}
\overrightarrow{X}_{S1} &= \overrightarrow{Y}_{S1} \wedge \overrightarrow{Z}_{S1} \\
&= \lambda.\left(\overrightarrow{Y}.\cos(\beta) + \overrightarrow{Z}.\sin(\beta)\right) \\
&\quad \wedge \left(\overrightarrow{X}.\cos(\beta).\sin(\alpha) - \overrightarrow{Y}.\cos(\alpha).\sin(\beta) + \overrightarrow{Z}.\cos(\alpha).\cos(\beta)\right) \\
&= \lambda.\left(\overrightarrow{X}.\cos(\alpha) + \overrightarrow{Y}.\cos(\beta).\sin(\beta).\sin(\alpha) - \overrightarrow{Z}.\cos(\beta)^2.\sin(\alpha)\right) \qquad (4.10)
\end{aligned}$$

4.2.3 Envoi de forces

Les forces sont exercées sur les trois degrés actifs à l'aide de trois petits moteurs de 20W de type RE25 développés par Maxon Motors ag. [Maxon Motor ag, 2002]. Les

moteurs sont reliés à l'outil par un système de réducteurs à câble d'usage classique en téléopération [Riwan, 2002] (voir figure 4.5). Le couple M_H de démarrage[14] des moteurs est de 240 mili Newton mètre[15] (mNm) pour une tension nominale U de 24 Volts, et 1 Volt de variation de cette tension fait varier ce couple de 10 mNm. Les tensions exercées sur les moteurs sont générées par des convertisseurs numériques analogiques (CNA) situés sur la carte PCI. Ces convertisseurs fournissent une tension proportionnelle à la valeur digitale qui leur est donnée en entrée. La tension maximale U_{max} reçue envoyée sur les moteurs par l'amplificateur vaut environ 3,5 Volts. Le couple maximal M_{max} exerçable sur chacun des deux axes X et Y vaut donc :

$$M_{max}(X) \;=\; M_{max}(Y) \;=\; M_H * \frac{U}{U_{max}} * \frac{r2}{r1} \;=\; 490\ mNm,$$

$r1$ et $r2$ représentent les diamètres des composantes des réducteurs (voir figure 4.5). La force maximale F_{max} applicable sur l'axe Z vaut :

$$F_{max}(Z) \;=\; M_H * \frac{U}{U_{max}} * \frac{1}{r1} \;=\; 7\ N,$$

En fait, on verra plus loin qu'on ne se préoccupe jamais de connaître la valeur exacte de la force émise. L'intérêt de cette étude est surtout de remarquer que la force émise est proportionnelle à l'ordre fourni à la carte.

4.3 Description de l'outil virtuel

4.3.1 Géométrie

L'outil virtuel est défini par le dilaté de rayon rad d'un segment de longueur l_{lie} appelé *squelette de l'outil*. L'extrémité de ce squelette est appelée *pointe de l'outil* et est noté $EndP$. À chaque pas de simulation t on a : $EndP = \mathcal{R}_{lie}.P + FixedP$ où $\mathcal{R}_{lie}$ est une matrice de rotation, P la position de l'outil telle qu'elle a été calculée plus haut à partir de la lecture des encodeurs, et $FixedP$ un point appelé *point fixe de l'outil*. Le duplet ($\mathcal{R}_{lie}$,$FixedP$) caractérise le positionnement au repos de l'outil virtuel dans la scène. On appelle Z_{lie} le vecteur unitaire de direction $FixedP - EndP$ (dirigé de $EndP$ vers $FixedP$), Y_{lie} le vecteur unitaire correspondant à l'axe d'ouverture des pinces ($Y_{lie} = \mathcal{R}_{lie}.\overrightarrow{A}$) et on pose $X_{lie} = Y_{lie} \wedge Z_{lie}$.

Dans certains cas, par exemple pour la manipulation des vaisseaux (cf. § 5.4), on a besoin d'une description plus précise de l'outil prenant en compte la géométrie de la pince. L'outil est alors décrit comme l'union des dilatés de quatre segments : le premier correspond au manche de l'outil, de rayon rad et d'extrémité $AEndP = EndP +$

14. Le couple de démarrage est le couple nécessaire pour maintenir l'axe arrêté lorsque l'on exerce sur les bornes du moteur sa tension nominale

15. Rappelons que le mili Newton mètre est le couple exercé par une masse d'environ 0,1 gramme placé au bout d'une barre de 1 mètre.

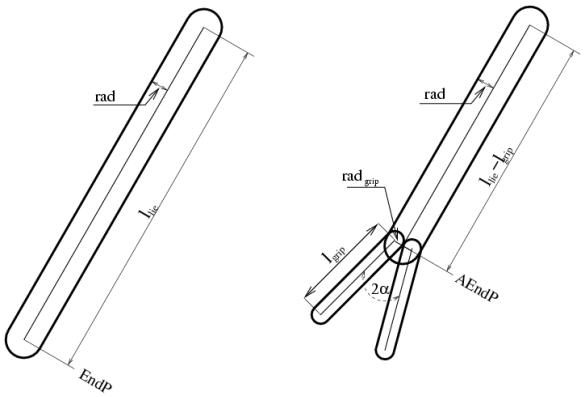


FIG. 4.6 – *Différentes modélisations de l'outil virtuel. A gauche pour les contacts avec le parenchyme et à droite pour les contacts avec les vaisseaux (cf. § 5.4).*

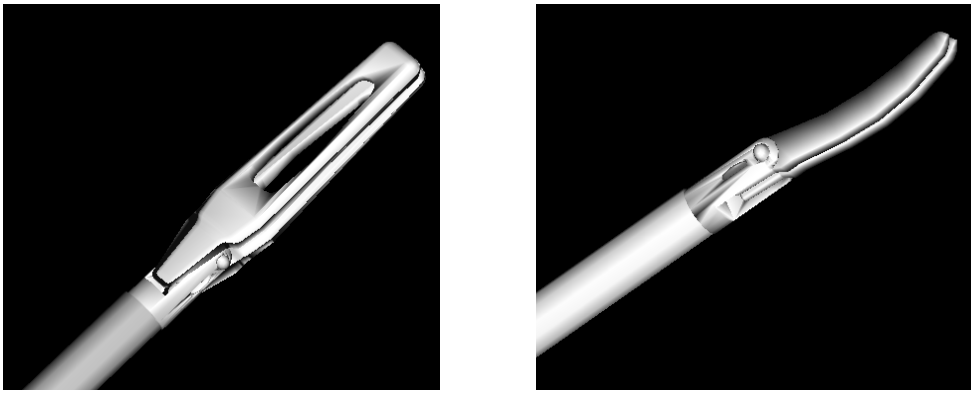


FIG. 4.7 – *Deux exemples de modèles de pinces laparoscopiques fournis par l'Ircad*

$l_{grip}.Z_{lie}$; les deux suivants, de rayon rad_{grip} et de longueur rad_{grip}, correspondent aux mâchoires de la pince. Leurs axes ont une direction égale à $\pm\sin(\alpha_{grip}).X_{lie} - \cos(\alpha_{grip}).Z_{lie}$ et s'appuient sur les points $AEndP \pm rad_{grip}.X_{lie}$; le dernier, de rayon rad_{grip}, relie la base des mâchoires au bout du manche. Ce dilaté est entièrement inclus dans les trois autres mais permet de disposer d'un squelette continu.

Ces modélisations sont bien sûr extrêmement simplifiées. La seconde ne différencie pas par exemple la largeur de l'épaisseur des mâchoires des pinces. La notion de dilaté les rend cependant très pratiques d'utilisation car savoir si une structure donnée intersecte l'outil revient à calculer la distance de cette structure par rapport aux segments composant le squelette de l'outil.

Des modèles volumiques de plusieurs des outils utilisés en chirurgie laparoscopique nous ont été transmis par l'IRCAD.

4.4 Traitement des contacts

L'étape qui suit la détermination de la position de l'outil et son positionnement dans la scène virtuelle est évidemment le traitement des collisions. Premièrement, il faut pouvoir détecter l'existence d'une collision, c'est-à-dire d'une interpénétration de l'outil virtuel et de l'une des structures de la scène. Ensuite, il faut décider quelles sont les actions à entreprendre pour résoudre cette pénétration.

4.4.1 Détection de collision

La détection de collision entre deux structures de forme quelconque est un problème crucial et difficile. La méthode naïve, qui consiste à mesurer la distance entre toutes les primitives (facettes) des deux structures a une complexité d'ordre (n^2) beaucoup trop grande pour pouvoir être appliquée en temps réel pour des objets un peu complexes. De nombreuses stratégies ont été développées pour résoudre ce problème, ces stratégies dépendant principalement de la manière avec laquelle sont représentés les objets en mémoire (CSG, fonctions implicites, polygones, etc.), des hypothèses formulées sur ces objets (simplicité de la géométrie, convexité, immobilité, etc.), et du type de question que l'on veut pouvoir résoudre (existence d'une intersection, localisation de cette intersection, distance minimum entre les objets, etc.).

Lin et Canny [Lin, 1993] ont proposé un algorithme permettant de calculer en temps quasi constant la distance entre deux solides mobiles mais convexes. L'algorithme mémorise à chaque pas la paire de primitives minimisant la distance entre ces deux objets et utilise le fait que la nouvelle paire est probablement à proximité de l'ancienne.

D'autres algorithmes dits de *décomposition hiérarchique* décomposent l'objet en un arbre de structures emboîtées. Ces structures peuvent être des sphères [Hubbard, 1996; Davanne *et al.*, 2002], des boîtes orientées le long des axes du repère (AABB trees) ou d'orientation quelconque (OBB trees) [Gottschalk *et al.*, 1996] ou encore d'autres formes géométriques relativement simples. L'intérêt de ces méthodes réside en la facilité du calcul de distance entre deux niveaux donnés de l'arbre, une structure étant d'autant plus puissante qu'elle nécessite moins de niveaux pour décrire un objet. Même si ces méthodes peuvent être étendues au cas d'objets déformables [van den Bergen, 1997] au prix d'une remise à jour dynamique la structure de données, elles ne sont que difficilement applicables lors de fortes déformations ou dans le cas de variation de topologie.

Hoff [Hoff *et al.*, 2002] propose un algorithme de détection de collision entre solides polygonaux, éventuellement déformables, de forme quelconque. L'intérêt de cet algorithme est de fournir un grand nombre de renseignements de proximité (volume intersecté, distance minimum, etc.), et cela sans nécessiter la maintenance d'une structure de données particulière et avec une complexité indépendante de la géométrie des objets. Bien qu'elle utilise certaines des capacités d'accélération fournies de manière

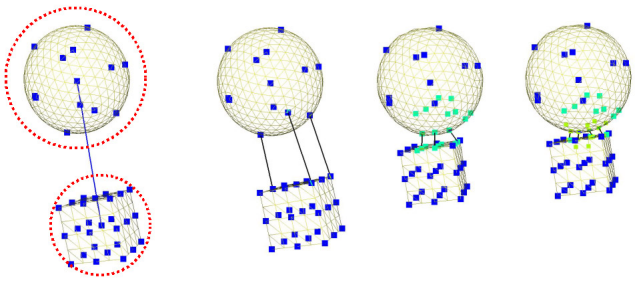


FIG. 4.8 – *Gestion multirésolution des collisions entre deux objets déformables. La couleur représente le niveau de discrétisation utilisé. (source [Imagis, 2001])*

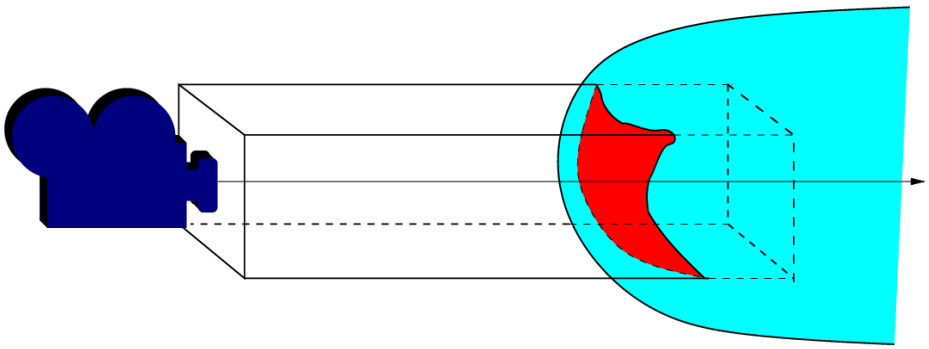


FIG. 4.9 – *Le champ de visualisation de la caméra correspond à la position de l'outil. En rouge les parties vues par la caméra, et donc en collision avec l'outil*

efficace par les cartes graphiques, cette méthode n'est pas suffisamment rapide pour être incluse dans un simulateur temps réel. Ce n'est pas le cas d'une autre méthode récemment mise en œuvre dans le projet iMagis [Imagis, 2001, p. 25] et qui utilise les maillages multirésolution présentés par Gilles Debunne [Debunne, 2000]. Les calculs sont faits de manière rapide pour une version simple des maillages et, au besoin, on accède aux maillages plus fins pour avoir des résultats plus précis (voir figure 4.8). Cette méthode n'a cependant pas fait l'objet de publication.

Pour le cas précis de la détection et la localisation des collisions entre un objet de forme géométrique relativement simple (l'outil virtuel) et un autre de forme quelconque et déformable (l'organe), une stratégie nommée LCN a été proposée par J.C.Lombardo [Lombardo *et al.*, 1999] dans le cadre de l'action AISIM. Cette stratégie utilise la très grande puissance des processeurs des cartes graphiques en y délocalisant les calculs. On utilise pour cela lae mode de sélection de la librairie OpenGL [Woo *et al.*, 1997, chap. 13]. L'idée est de définir une caméra dont le champ de visualisation a exactement la forme de l'outil dont on cherche à détecter les collisions. On affiche alors dans cette caméra les autres objets de la scène. Les triangles effectivement affichés sont alors exactement les triangles de la surface en intersection avec l'outil. Il est possible de prendre en compte les mouvements de l'outil en utilisant une caméra dont le champ de visualisation correspond au volume balayé par l'outil entre les deux pas de temps. Une telle technique n'est malheureusement pas aisée à mettre en œuvre pour

les mouvements de l'objet. Compte tenu d'une part de l'amplitude des mouvements relatifs des outils et du parenchyme et d'autre part de la géométrie et de la taille de ce dernier, on supposera qu'une collision au cours d'un pas de temps se traduira toujours par une collision à la fin de ce pas de temps.

Avec le matériel actuel, une détection de collision s'effectue entre un peu moins d'un millième de seconde et deux millièmes de seconde, en fonction du nombre de triangles de la surface du maillage. L'une des étapes les plus coûteuses de cette méthode est en effet la transmission à la carte de la géométrie du maillage que l'on cherche à tester. Le problème pour un objet déformable est qu'il faut transmettre ces données à chaque itération. On a cherché à limiter de plusieurs façons le temps nécessaire à cette transmission, par exemple en séparant l'envoi de la topologie du maillage et celui de la position de ses sommets (technique dite du *vertexArray*, cf. § 6.4.3) ou en réutilisant la même géométrie pour chacun des outils virtuels (technique des *display-list*). Les essais ne se sont malheureusement pas révélés fructueux. Dans l'état actuel, les performances de cette méthode nous semblent cependant assez satisfaisantes pour l'usage que nous en avons, l'un de ses intérêts étant qu'elle ne nécessite pas l'usage d'une structure de données particulière. Récemment, Aharon [Aharon and Lenglet, 2002] a proposé une méthode utilisant une hiérarchie simplifiée de boîtes orientées pour filtrer les triangles fournis à la carte graphique et ainsi accélérer la détection de collision. Cette méthode n'a pas encore été testée pour le simulateur.

4.4.2 Principe

Le traitement des collisions entre objets non interpénétrables est une problématique vaste possédant de nombreuses applications dans des domaines aussi variés que l'automobile, la mise en forme de matériaux, la modélisation de vêtements, l'animation, etc. On peut distinguer deux grandes classes de méthodes.

Les méthodes par contraintes [Witkin *et al.*, 1994; Baraff, 1993] résolvent ce problème en divisant l'instant t_c où se produit le choc en deux instants distincts t_c^- et t_c^+. La simulation est interrompue à l'instant t_c^- et reprise à l'instant t_c^+ en modifiant les vitesses des objets et en imposant de nouvelles contraintes dans le mouvement. Ces stratégies ne sont évidemment pas applicables à notre cas car elles nécessitent pour un même pas de temps d'effectuer plusieurs fois les calculs, et ce d'autant plus souvent qu'il y a de chocs entre les objets. De plus, le calcul des impulsions, nécessaire lors du passage entre les instants t_c^- et t_c^+, peut se révéler lourd et coûteux. Elles ne permettent donc pas le temps réel.

Les méthodes par pénalité [Deguet *et al.*, 1998] consistent à introduire une force $F = \lambda.x$ proportionnelle à la pénétration des objets de façon à inciter leur éjection. Ces méthodes sont très populaires car très simples à mettre en œuvre. Elles sont utilisées dès que le critère temps est important, par exemple pour les jeux vidéos. Elles ont cependant le défaut de ne ne pas garantir la non pénétration des objets. Une masse ponctuelle subissant une force de type gravité et posée sur un plan le pénétrera de telle manière que la force F compense exactement son poids. Un autre défaut de

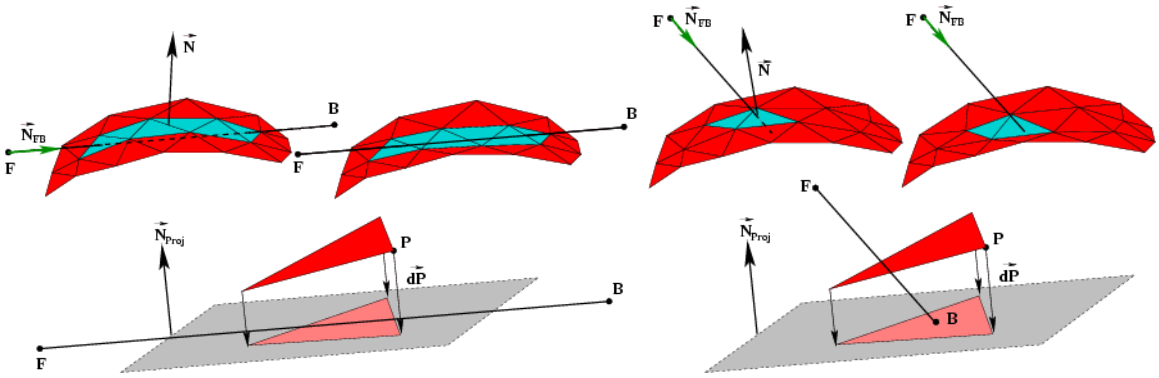


FIG. 4.10 – *Définition du plan de projection et des faces intersectées sur celui-ci pour l'ancienne méthode de traitement des collisions [Picinbono, 2001].*

ces méthodes est la difficulté d´évaluation du coefficient λ. Une valeur trop faible va causer de profondes interpénétrations alors qu'un valeur trop grande peut rejeter trop loin les objets, surtout si d'autres forces s'exercent simultanément sur ceux-ci.

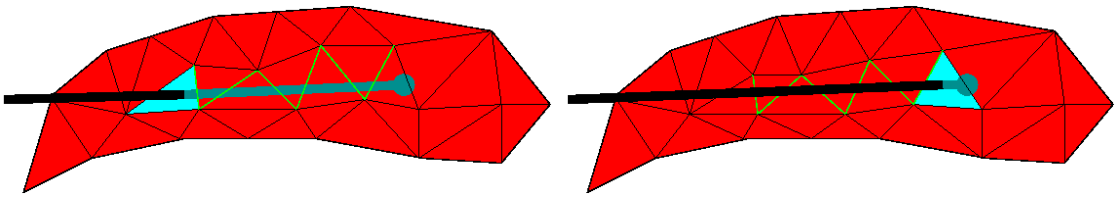
La méthode que nous employons cherche à garantir la non pénétration du maillage du parenchyme par l'outil virtuel. Elle consiste, lorsqu'une collision est détectée, à modifier directement la position des sommets du maillage de façon à le sortir de l'outil.

Les versions précédentes du simulateur [Picinbono *et al.*, 2001a] différencient deux types de contacts entre l'outil et le maillage. Le premier est un contact ponctuel et est réalisé avec la pointe de l'outil. On calcul la normale moyenne $\overrightarrow{N}$ des triangles au voisinage de cette pointe puis on repousse ces triangles sur le plan défini par ce vecteur $\overrightarrow{N}$ et la pointe de l'outil. Le deuxième type de contact est un contact ligne, réalisé avec le manche de l'outil. Les triangles du voisinage sont alors repoussés sur un plan perpendiculaire à celui défini par l'axe de l'outil et la normale moyenne $\overrightarrow{N}$ de ces triangles (voir figure 4.10). Les voisinages de la pointe et du manche sont obtenus grâce à deux détections de collision successives.

Cette manière de procéder fonctionne très bien lorsque la surface de l'organe est lisse et qu'il n'y a qu'une zone de contact pour le manche. En effet, la valeur de la moyenne des normales des triangles du voisinage n'a de sens que si l'orientation de ces triangles présente une certaine cohérence. Malheureusement cette cohérence n'existe plus pour les zones déjà découpées et la méthode précédente n'est plus applicable des que l'on veut agir sur ces zones, par exemple lorsque l'on cherche à écarter avec le manche de l'un des outils l'un des segments découpés pour laisser plus de champ de vision. On a donc été conduit à développer une nouvelle méthode. Celle-ci fonctionne en plusieurs étapes (voir figure 4.11) :

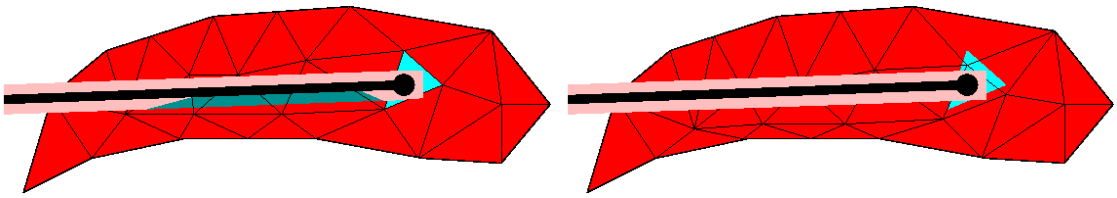
(a) d'abord on détermine quelles sont les arêtes ayant franchi le squelette de l'outil au cours du pas de temps précédent. Pour cela on commence par teste les arêtes du triangle intersectant le squelette de l'outil et on procède de proche en proche.

(b) on empêche alors ces arêtes de franchir le squelette de l'outil. Pour cela, on



(a) En bleu le triangle intersectant le squelette de l'outil. En vert les arêtes ayant intersecté ce squelette au cours du pas de temps précédent.

(b) Une fois les arêtes repoussées seul le triangle (en bleu) dans lequel est passé la pointe de l'outil est encore éventuellement pénétré par ce dernier.



(c) Le squelette de celui-ci est maintenant totalement hors du maillage. Son volume (en rose) est par contre encore en contact.

(d) Le volume de l'outil est maintenant totalement hors du maillage.

FIG. 4.11 – *Les grandes étapes de notre méthode de traitement des collisions*

bloque leur mouvement relativement au squelette.

(c) ensuite, on finit de repousser le maillage hors du squelette de l'outil en déplaçant les triangles situés à proximité de la pointe.

(d) enfin, on repousse arêtes et sommets du maillage hors du volume de l'outil en les éloignant du squelette de celui-ci d'une distance suffisante.

4.4.3 Éjection du squelette de l'outil

Afin de tenir compte simultanément du mouvement du maillage et du mouvement de l'outil, on rapporte toutes les positions dans le référentiel lié à l'outil. On suppose de plus qu'entre deux pas de temps, les mouvements des sommets du maillage sont linéaires dans ce référentiel. Cette dernière approximation est différente de celle qui estime que les mouvements des sommets sont linéaires dans le repère de la scène et elle n'est valable que si la composante de rotation du mouvement de l'outil est suffisamment faible. Dans le cas contraire certaines intersections ne seront pas détectées (voir figure 4.12). Dans le cadre du simulateur, les mouvements de l'outil étant effectivement faibles, l'approximation reste satisfaisante.

On considère que l'outil virtuel a subi, entre les deux pas de simulation $t-1$ et t, une transformation $\mathcal{R}$ transformant d'une part le vecteur $Z_{lie}(t-1)$ en Z_{lie} et d'autre part le point $EndP(t-1)$ en $EndP$, et qui laisse inchangé le vecteur $Z_{lie}(t) \wedge Z_{lie}(t-1)$.

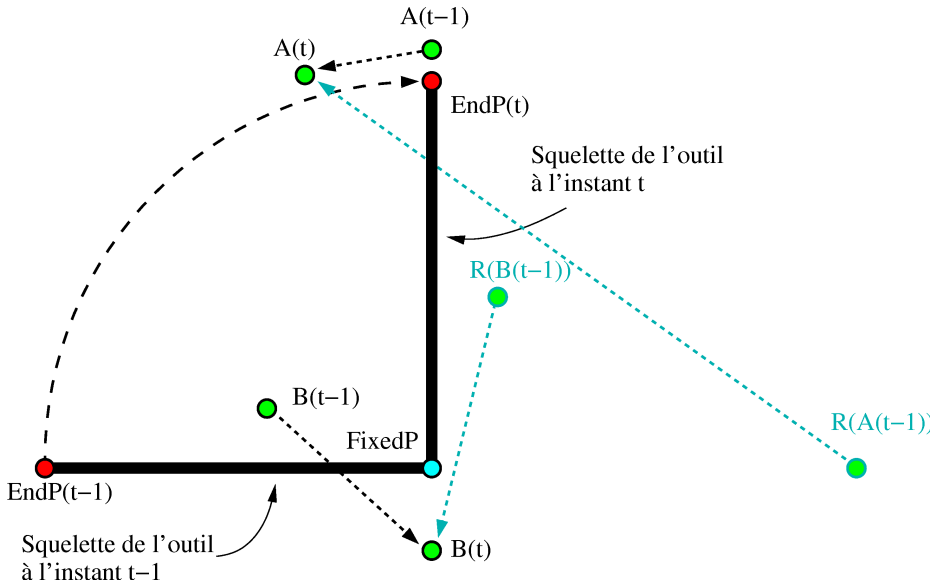


FIG. 4.12 – *Exemple de déplacement de l'outil rendant invalide l'approximation du changement de référentiel. Le point A sera détecté comme intersectant l'outil mais pas le point B.*

Dans le cas où les vecteurs $Z_{lie}(t-1)$ et Z_{lie} sont égaux, $\mathcal{R}$ est égale à la translation de vecteur $EndP(t) - EndP(t-1)$:

$$\begin{aligned} \mathcal{R}\Big(Z_{lie}(t-1)\Big) &= Z_{lie}(t) \\ \mathcal{R}\Big(EndP(t-1)\Big) &= EndP(t) \\ \mathcal{R}\Big(Z_{lie}(t-1) \wedge Z_{lie}(t)\Big) &= Z_{lie}(t-1) \wedge Z_{lie}(t) \end{aligned}$$

Détermination des arêtes ayant intersecté l'outil

Soit E une arête du maillage, $\mathcal{R}^{-1}(A)$ et $\mathcal{R}^{-1}(B)$ la position de ses sommets à la fin du pas de temps précédent et $A + \Delta A$ et $B + \Delta B$ leur position actuelle. Compte tenu des approximations précédentes, savoir si l'arête E intersecte ou non le squelette de l'outil au cours du pas de temps revient à trouver un triplet (t,α,δ) dans $[0,1] \times [0,1] \times [0,l]$ vérifiant :

$$A + t.\Delta A + \alpha.(B + t.\Delta B - A - t.\Delta A) = EndP + \delta.Z_{lie} \tag{4.11}$$

Pour résoudre cette équation, on détermine les instants t au cours desquels la droite support de l'arête E intersecte l'axe de l'outil et on vérifie si cette intersection correspond bien à une intersection de l'arête. Pour trouver les instants t d'intersection, on commence par prendre le produit vectoriel de l'égalité précédente avec le vecteur

$$(B + t.\Delta B - A - t.\Delta A) \wedge Z_{lie}$$

de façon à supprimer les termes en α et en δ et obtenir ainsi une équation du second degré d'inconnue t :

$$\begin{aligned}
&t^2.\Big(\Delta A.((\Delta B - \Delta A) \wedge Z_{lie})\Big) \\
&+t.\Big(\Delta A.((B - A) \wedge Z_{lie}) - (EndP - A).(\Delta B - \Delta A) \wedge Z_{lie})\Big) \\
&= (EndP - A).((B - A) \wedge Z_{lie})
\end{aligned}$$

Soient t_1 et t_2 les deux racines de l'équation précédente. Ces deux racines correspondent aux deux instants où les axes de l'arête et de l'outil se croisent. Pour savoir si ces instants correspondent réellement à une intersection de l'arête avec le squelette de l'outil il faut calculer les valeurs α_1, α_2, δ_1 et δ_2 correspondantes. Pour cela, on injecte les valeurs t_1 et t_2 dans l'équation 4.11. On calcule alors d'une part α_1 et α_2 en prenant le produit vectoriel de cette équation avec le vecteur Z_{lie} de façon à supprimer le terme en δ et d'autre part $\delta 1$ et δ_2 en prenant le produit vectoriel avec le vecteur $(B + t_{1\backslash 2}.\Delta B - A - t_{1\backslash 2}\Delta A)$ de façon à supprimer le terme en α.

$$\begin{aligned}
\alpha_1 &= \frac{(EndP - A - t_1\Delta A) \wedge Z_{lie}}{(B + t_1\Delta B - A - t_1\Delta A) \wedge Z_{lie}} \\
\delta_1 &= \frac{(A + t_1\Delta A - EndP) \wedge (B + t_1\Delta B - A - t_1\Delta A)}{Z_{lie} \wedge (B + t_1\Delta B - A - t_1\Delta A)}
\end{aligned}$$

Un triplet (t_i,α_i,δ_i) correspond effectivement à un franchissement du squelette de l'outil par l'arête s'il appartient à l'ensemble [0,1] × [0,1] × [0,1]. Si exactement l'un des deux triplets correspond à un franchissement du squelette de l'outil par l'arête, on dit que l'arête a *franchi le squelette de l'outil*. Dans le cas contraire, on dit qu'elle *ne l'a pas franchi*. En effet, si les deux triplets correspondent à un franchissement, cela signifie que l'arête a d'abord franchi l'outil mais est ensuite repassée dans l'autre sens.

Supposons qu'à la fin du pas de temps précédent, le squelette de l'outil ne pénètre pas le maillage. En supposant de plus que l'outil n'a pas traversé entièrement le maillage, une collision entre le maillage et l'outil pendant le pas de simulation entraîne donc une pénétration du maillage par l'outil à la fin de ce pas. Du fait de la géométrie du foie et de l'amplitude des mouvements de l'outil, cette dernière hypothèse est effectivement vérifiée. On détermine les triangles percés par le squelette de l'outil à l'aide de la méthode de détection de collision présentée précédemment (cf. § 4.4.1). On teste alors le franchissement des arêtes, en débutant par celles des triangles intersectés et en procédant de proche en proche (voir figure 4.11 (a)). On pourrait se passer de l'hypothèse sur l'amplitude des mouvements relatifs du maillage et de l'outil mais il faudrait alors tester l'ensemble des arêtes de la surface du maillage, ce qui serait beaucoup trop long.

Déplacement du maillage hors de l'outil On cherche à modifier la position des sommets du maillage de façon à ce qu'aucune arête n'intersecte le squelette de l'outil. Les premiers essais que nous avons faits consistaient à déterminer, pour chacun

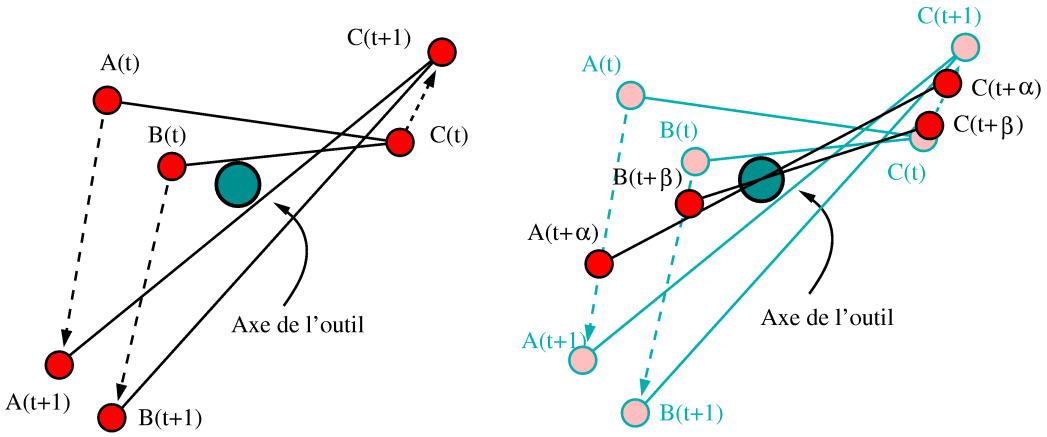


FIG. 4.13 – *Vue axiale du référentiel de l'outil entre les temps de simulation t et $t+1$. L'arête AC franchit le squelette de l'outil en $t+\alpha$, l'arête BC en $t+\beta$ ($\beta < \alpha$). Le positionnement des sommets au premier instant d'intersection ($t+\beta$) causerait le franchissement de l'arête AC.*

des sommets proches de l'outil, l'instant où la première de ses arêtes concourantes intersecte le squelette de l'outil, puis à bloquer, dans le référentiel de l'outil, le mouvement des sommets à cet instant. Malheureusement, pour certaines configurations, cette stratégie relativement simple n'empêche pas le franchissement du squelette. La figure 4.13 illustre un tel type de configuration.

Pour chaque sommet adjacent à une arête ayant croisé le squelette de l'outil, on maintient donc un *intervalle d'acceptabilité* $[t_{min},t_{max}]$, défini de telle manière que, quelles que soient les positions prises par les sommets à l'intérieur de ces intervalles, aucune arête n'ait croisé le squelette de l'outil. Ces intervalles sont calculés de la manière suivante.
Les intervalles sont initialisés à [0,1]. Soit $E = (A,B)$ une arête du maillage croisant le squelette de l'outil à l'instant t_c. Soit ΔA le déplacement du sommet A au cours du dernier pas de temps **dans le référentiel de l'outil**. Si le mouvement du sommet A approche l'arête E du squelette de l'outil, alors on prend $[t_{min},t_{max}] \cap [0,t_c]$ comme nouvelle valeur pour l'intervalle de ce sommet. S'il l'en éloigne, on prend $[t_{min},t_{max}] \cap [t_c,1]$. Pour déterminer si le mouvement du sommet approche ou éloigne l'arête, il suffit de considérer le signe de $\Delta A \times Zd.((EndP - A) \times Zd)$ avec $zd = Z \wedge AB$.
Dans le cas où l'intervalle d'acceptabilité de l'un des deux sommets de l'arête se révélerait être vide, on immobilise ce sommet à la position correspondant au t_{min} de son précédent intervalle d'acceptabilité s'il approche l'arête du squelette de l'outil, et à t_{max} sinon. On détermine alors le nouvel instant $t_c^{'}$ de collision de l'arête et on met à jour l'intervalle d'acceptabilité de l'autre sommet. L'expérience montre que le nouvel intervalle d'acceptabilité ainsi déterminé n'est jamais vide.
Lorsque tous les intervalles sont déterminés, les sommets sont déplacés à la position correspondant à leurs t_{min}.

À la fin de cette étape, les arêtes étant entrées en collision avec le squelette de

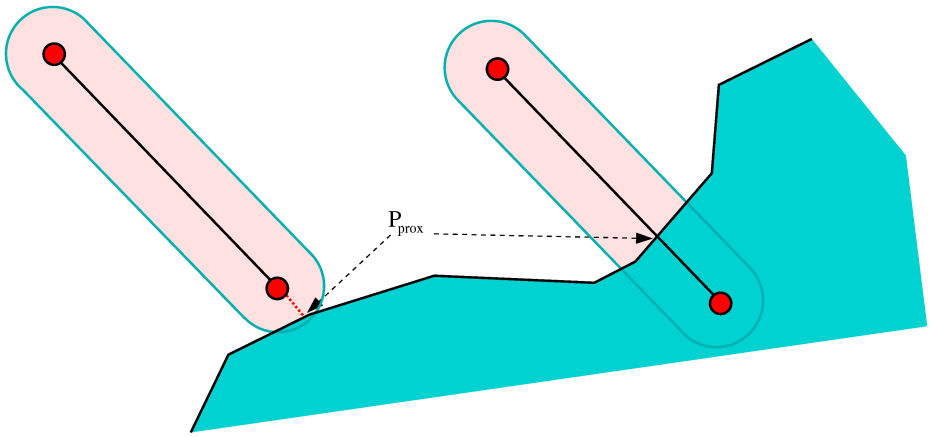


FIG. 4.14 – *Définition de P_{prox}*

l'outil ont toutes été repoussées. Cependant, si la pointe de l'outil était à l'intérieur du maillage, elle y reste.

4.4.4 Déplacement des triangles proches de la pointe

L'étape suivante est le déplacement des triangles situés au voisinage de la pointe de l'outil hors du volume de l'outil.

Soit $\mathbf{P}_{prox}$ le point de la surface du maillage le plus proche de la pointe de l'outil. Si le squelette de l'outil intersecte le maillage on place $\mathbf{P}_{prox}$ à l'intersection (cf. figure 4.14). On définit alors la normale $\mathbf{n}_{prox}$ en ce point par $\mathbf{n}_{prox} = \alpha_0.n_o + \alpha_1.n_1 + alpha_2.n_2$ où α_0, α_1 et α_2 sont les coefficient barycentriques de $\mathbf{P}_{prox}$ dans le triangle auquel il appartient et n_0, n_1 et n_2 les normales des sommets correspondants.

Une fois le vecteur n déterminé, on repousse ce triangle sur le plan $\mathcal{P}$ défini par ce vecteur et situé à une distance de la pointe de l'outil égale au rayon de celui-ci. De plus, pour éviter les effets dus à un passage d'un triangle à un triangle voisin, on va aussi repousser, mais de manière moindre, les sommets voisins des sommets du triangle. À chaque sommet on affecte un coefficient. Si ce sommet est adjacent à un seul sommet du triangle, ce coefficient est égal au α_i correspondant défini plus haut. S'il est adjacent à deux sommets, il est égal à la somme des deux α_i correspondant (voir figure 4.15). On déplace alors les sommets vers le plan $\mathcal{P}$ proportionnellement à ce coefficient. À la fin de cette étape, le squelette de l'outil n'intersecte plus le maillage.

Cette étape considère que l'ensemble des triangles situés à proximité de la pointe constituent une unique composante connexe. Il peut arriver, lorsque cela n'est pas vérifiée, que des comportements non désirés apparaissent du fait de la projection de triangles supplémentaires sur le plan $\mathcal{P}$.

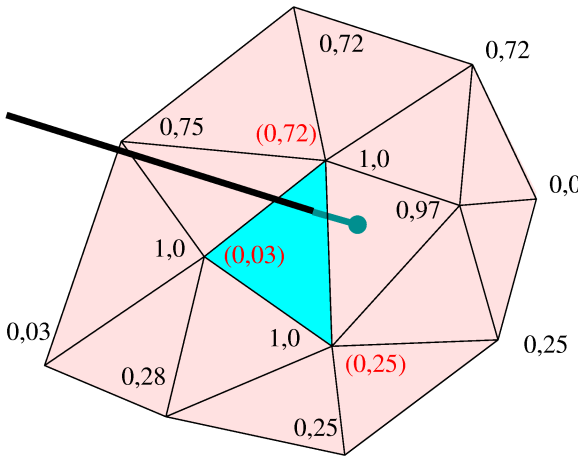


FIG. 4.15 – *Pour sortir le maillage de l'outil, on repousse les sommets voisins de la pointe proportionnellement à un coefficient (en noir) calculé à partir des coordonnées barycentriques (en rouge) de l'intersection du squelette et du maillage par rapport aux sommets du triangle intersecté.*

4.4.5 Déplacement des arêtes et des sommets

La dernière étape consiste simplement à repousser le maillage en dehors du volume de l'outil. On commence par repousser les sommets, puis on repousse les arêtes. On peut remarquer que le déplacement d'un sommet perpendiculairement au squelette de l'outil ne causera jamais le franchissement de celui-ci par l'une des arêtes concourantes à ce sommet. Au contraire, le déplacement d'une arête peut très bien causer le franchissement de l'axe d'une autre arête qui lui est concourante. De tels cas correspondent cependant à de très fortes déformations du maillage et ne se rencontrent pas en pratique.

4.4.6 Prise et Découpe

La fermeture de la pince virtuelle sur le maillage peut générer, suivant le type de l'outil, deux comportements différents. Si l'outil est en mode pince, on n'effectue pas la deuxième étape du traitement des contacts qui consiste à déplacer les triangles du maillage proches de la pointe. À la place, on mémorise la position relative des sommets du triangle de la surface qui contient le point P_{prox} par rapport à l'outil. Lors des itérations suivantes, et tant que la pince est maintenue fermée, ces sommets seront repositionnés de façon à être immobiles par rapport à l'outil. Lorsque la pince s'ouvre, le traitement des contacts reprend son cours normal.

Si l'outil est en mode bistouri électrique, la fermeture de la pince doit causer la suppression d'un certain nombre de tétraèdres. On commence par répertorier l'ensemble des tétraèdres *proches* de l'outil. Par proche de l'outil, on entend situés à moins de $2.rad$ de son axe. On vérifie ensuite que la taille de chacune des arêtes de ces tétraèdres est bien inférieure à cette même longueur rad. Dans le cas contraire,

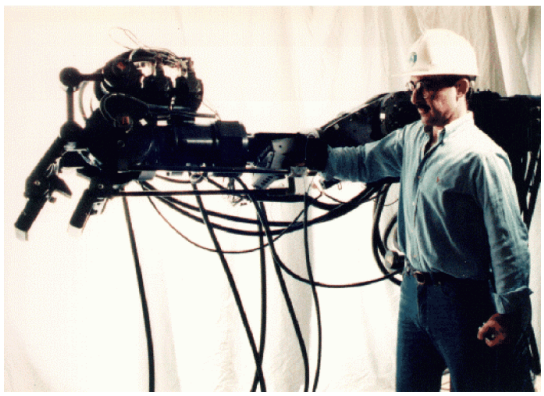
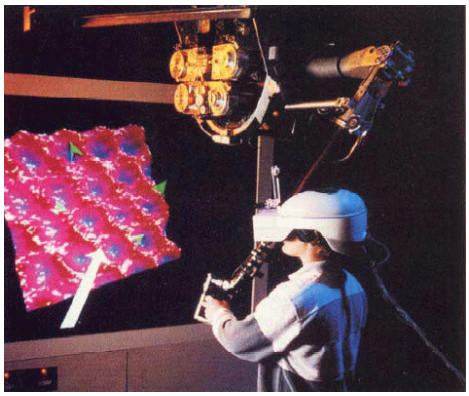



FIG. 4.16 – *Gauche : exemple d'un extenseur mécanique à 6 degrés de libertés proposé par le professeur H. Kazerooni (source* `www.me.berkeley.edu`*). Droite : le nanomanipulateur développé à l'université de Chapel Hill (source [Taylor* et al.*, 1993])*

on procède à un ou plusieurs raffinements successifs de la zone concernée. On essaie alors de retirer l'ensemble des tétraèdres situés proches de l'outil et appartenant à la surface du maillage. Ce processus est décrit avec plus de précision dans la partie 3.10.3.

4.5 Retour d'effort

4.5.1 Historique et généralités

La problématique du retour d'effort provient du domaine de la téléopération. Les premiers systèmes de téléopération ont été développés au milieu des années 50, dans l'industrie nucléaire, pour permettre aux opérateurs de manipuler des substances dangereuses sans se mettre en danger. Pour ces premiers systèmes, les bras maîtres et esclaves étaient couplés mécaniquement à l'aide de câbles et de poulies. Les progrès de l'électronique et de l'automatique dans les années 60, puis ceux de l'informatique dans les années 80, ont permis de séparer les bras maîtres et esclaves, le couplage devenant analogique, puis numérique, rendant possible des applications dans des domaines aussi variés que la téléopération spatiale, médicale, ou sous-marine. Malheureusement, avec l'abandon du couplage mécanique, l'opérateur perd l'ensemble des informations haptiques[16] dont il disposait précédemment. Ces informations sont pourtant importantes pour un grand nombre d'applications et elles permettent d'opérer de manière plus efficace et plus sûre. Par exemple, elles permettent d'informer rapidement sur la prise d'un objet ou l'établissement d'un contact (par exemple dans le cas des nanomanipulateurs). Elles permettent aussi de restreindre l'amplitude des mouvements et d'éviter l'envoi d'ordres pouvant conduire à la rupture du matériel (dans le cas des extenseurs mécaniques, voir figure 4.16). Dans le cadre d'un simulateur de chirurgie, l'intérêt du

16. Le terme *haptique* vient du grec *αφή*, *haphtésaï*, qui signifie sentir par le toucher.

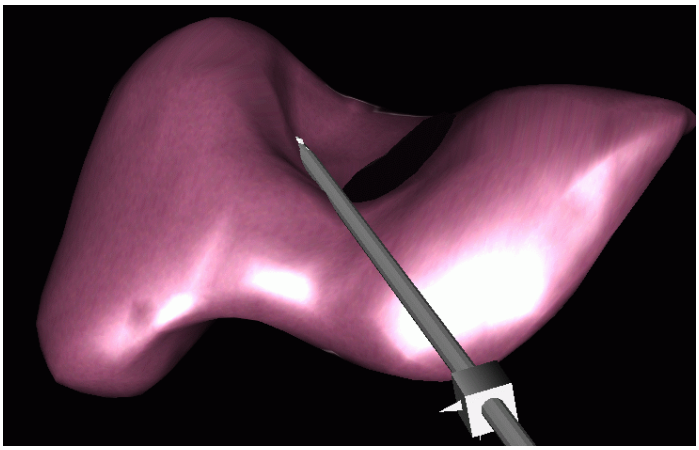


FIG. 4.17 – *Un mouvement non contraint par le retour d'effort peut entraîner des déformations aberrantes*

retour d'effort peut paraître paradoxal : les frottements des pinces sur les trocarts étant très importants, les chirurgiens n'ont pas vraiment le sentiment de sentir l'organe lors de l'opération. Pourtant, son absence conduit immanquablement à des déformations visuellement peu crédibles (voir figure 4.17). Son importance s'est donc ressentie très tôt et elle constitue un domaine actif de la recherche appliquée [Davanne *et al.*, 2002; Maaß *et al.*, 2003; Petersik *et al.*, 2003].

On peut distinguer chez l'homme deux types d'informations reçues lors du toucher ou de la manipulation d'un objet [Kheddar, 2002] :

- les informations de *kinesthésie*, qui décrivent les positions relatives et le mouvement de l'objet, ainsi que les efforts musculaires,
- les informations *tactiles*, qui décrivent les échanges se produisant entre la peau et l'environnement : forme, distribution spatiale des petites forces, échanges thermiques, rugosité, etc.

Si la plupart des systèmes à retour d'effort existants se préoccupent uniquement de kinesthésie, il existe des systèmes à retour tactile. Le CyberTouch par exemple, vendu par la société Immersion Inc., est un gant disposant de petits stimulateurs situés sur chaque doigt et permettant simuler certaines sensations simples telles que des pulsations ou des vibrations. Le docteur Harald Fischer de l'Université de Karlsruhe a développé un système tactile constitué de 64 aiguilles pouvant être activées individuellement. Ce système est conçu pour être utilisé avec des forceps endoscopiques et permet de transmettre les informations de pression détectées par des senseurs placés entre les mâchoires du forceps. Une énumération un peu ancienne mais très complète des différents types d'interfaces à retour d'effort peut être consultée dans [Youngblut *et al.*, 1996].

Un utilisateur donné peut percevoir des variations de position à une fréquence d'environ 20 à 30 Hz, même si cette fréquence varie sensiblement avec l'intensité de l'effort [Kheddar, 2002; Gosselin, 2000; Ellis *et al.*, 1997]. D'autre part, s'il est

FIG. 4.18 – *Gauche : le CyberTouch de Immersion. (source* www.immersion.com*) Droite : le système tactile développé par le docteur H. Fischer. (source* www.fzk.de*)*

possible de ressentir des vibrations jusqu'à 10000 Hz, deux signaux distincts ne sont différenciables que jusqu'à environ 300 Hz. Ces valeurs imposent donc un certain certain nombre de caractéristiques minimales pour les systèmes à retour d'effort : les consignes d'effort devront être générées au minimum à une fréquence de 300 Hz pour ne pas générer de vibrations et les pas de simulation devront être effectués à une fréquence d'au moins 20 Hz pour ne pas être ressentis.

4.5.2 Solutions précédentes

Solide rigide

Les premiers articles se préoccupant du retour d'effort s'intéressaient principalement à l'interaction entre un outil ponctuel et une surface rigide relativement lisse. Les solutions mises en œuvre introduisent la notion de *proxy* [Zilles and Salisbury, 1995; Ruspini *et al.*, 1997]. Le proxy représente la position apparente de l'outil dans la scène par opposition à sa position réelle qui est, elle, déduite directement de la position physique du périphérique. Si l'outil est situé à l'extérieur de l'objet, la position du proxy et celle de l'outil sont identiques. Si la trajectoire de l'outil pénètre l'intérieur de l'objet, le proxy se positionne sur la surface de celui-ci à l'intersection de cette trajectoire (voir figure 4.19). Le proxy se déplace alors sur la surface de façon à se rapprocher au maximum de la position réelle de l'outil. La force envoyée à l'utilisateur est alignée avec la différence entre les positions de l'outil et du proxy et son intensité est proportionnelle à la distance séparant ces deux positions (une fois ramenées dans le référentiel physique du périphérique). Il s'agit donc d'une force ressort de type $F = k.\Delta x$. En jouant sur le coefficient de raideur k, on peut faire varier la rigidité des objets. On peut même simuler des objets collants en forçant pendant un temps le maintien du proxy sur la surface de l'objet lorsque l'outil en sort.

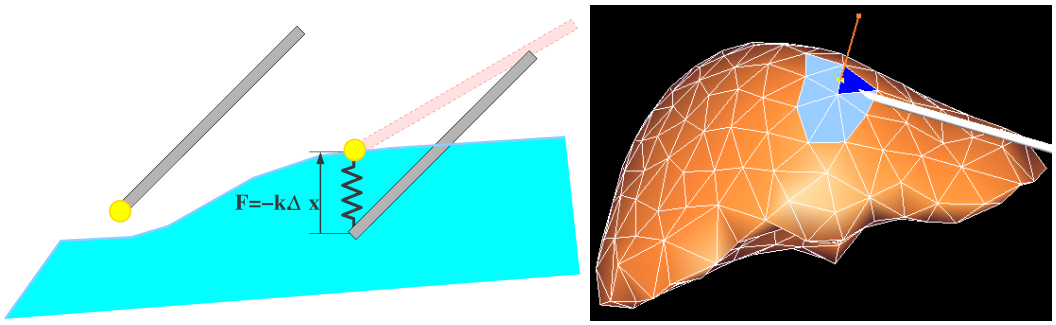


FIG. 4.19 – *Gauche : lorsque l'outil pénètre le maillage, le proxy (en jaune) reste sur la surface. Droite : exemple d'interaction entre le PhantoM et un maillage de foie. La position du proxy est représentée ainsi que la direction et l'intensité de la force correspondante. En pointillé, on représente l'outil tel qu'il peut être affiché à l'écran de manière à simuler la non pénétration.*

Calcul des forces pour un solide déformable

Il est probablement possible d'étendre la méthode précédente aux cas des contacts avec un solide déformable lorsque ceux-ci sont traités à l'aide d'une méthode de type pénalité (cf. § 4.4). Il s'agit simplement d'exercer, sur les sommets voisins du proxy, une force proportionnelle et opposée à celle envoyée sur l'outil. Pour éviter que, visuellement, l'outil pénètre le maillage, il est possible de le représenter non à sa position réelle, mais à la position de son proxy. Le choix du coefficient de proportionnalité entre la force exercée sur l'outil et celle exercée sur le maillage se révèle cependant difficile. Une autre difficulté de cette méthode est qu'elle n'est applicable en l'état qu'à des outils ponctuels. Pour des outils tels que le LIE, le contact peut en effet s'effectuer à la fois sur le manche et sur la pointe ce qui conduit à introduire plusieurs proxies, un par composante connexe du contact.

La méthode de traitement des collisions que l'on utilise consiste à déplacer les sommets du maillage de manière à positionner ce dernier en dehors du volume de l'outil. Comme il n'y a plus de notion de proxy, se pose la question du calcul de la force à renvoyer à l'utilisateur. On a d'abord pensé renvoyer une force proportionnelle aux forces internes des sommets qui ont été déplacés pour sortir le maillage de l'outil. Le problème qui se pose est que ce coefficient de transmission est difficile à calibrer et qu'il est de plus très dépendant du type de force interne utilisé pour le calcul des déformations. On a finalement renoncé à envoyer une force directement liée aux forces internes. L'option retenue pour les versions précédentes du simulateur consiste à émettre une force proportionnelle au déplacement subi par le maillage. Pour simplifier, on peut dire que la force a la direction de la normale moyenne du maillage au voisinage du contact avec l'outil et une intensité proportionnelle au plus grand des déplacements générés par la sortie du maillage de l'outil. Dans le cas où l'outil virtuel touche le maillage à la fois par le manche et par la pointe, la force envoyée est la somme des deux forces.

Envoi des forces pour un solide déformable

On a vu précédemment que pour pouvoir simuler des contacts mous sans que l'utilisateur ait de sensation d'à-coups ou de vibrations, il faut que le simulateur génère des consignes de force à une fréquence au moins égale à 300 Hz. Comme les calculs de déformations sont, à eux seuls, beaucoup trop longs pour être effectués en moins de 0,003 secondes, l'envoi des forces a dû en être découplé. Un processus particulier, fonctionnant à une fréquence suffisante, est donc créé et est chargé du calcul des forces. Ce processus dialogue avec le processus de calcul des déformations à l'aide d'un système de *socket* ou encore de segments de mémoire partagée.

La librairie GHOST [Sen, 1997] fournie par Sensable Technologies pour utiliser le PhantoM permet de mettre en œuvre facilement ce concept. Le programmeur peut définir une fonction chargée d'évaluer les forces transmises à l'utilisateur, fonction qui sera itérée à 1000Hz. Pour le LIE de Immersion, il n'existe malheureusement pas de telle bibliothèque et chaque développeur doit recoder son propre protocole de communication.

Lors de l'action AISIM, ont été proposées plusieurs méthodes permettant de générer des forces à une fréquence élevée [Picinbono and Lombardo, 1999; Picinbono *et al.*, 2001a] de manière à peu près satisfaisante. À chaque pas de temps t_n, le processus calculant les déformations estime la valeur de la force $\mathbf{F}_n$ exercée sur l'outil. Cette valeur est ensuite transmise au processus chargé de l'envoi des forces à l'utilisateur. La force effectivement émise à l'instant t est obtenue en extrapolant les forces $\mathbf{F}$ reçues. Les deux méthodes d'extrapolation proposées sont l'extrapolation linéaire en temps et celle linéaire en position. C'est cette dernière qui se révèle être la plus efficace, car limitant le mieux l'erreur entre la force calculée $\mathbf{F}(t_{n+1}^-)$ et la force effectivement reçue $\mathbf{F}_{n+1}$ (voir figure 4.20). Cette erreur se traduit en effet par une discontinuité qui peut se révéler sensible. Le principe de cette extrapolation en position est d'estimer la force émise sur l'outil à l'instant actuel t en fonction, d'une part, des deux précédentes forces $\mathbf{F}_{n-1}$ et $\mathbf{F}_n$ fournies par le processus de calcul des déformations, des positions de l'outil ($\mathbf{P}_{n-1}$ et $\mathbf{P}_n$) et, d'autre part, de la position actuelle de l'outil à l'instant t ($\mathbf{P}(t)$).

$$\forall t \in [t_{n-1},t_n],\ \mathbf{F}^{pos}(t) = \mathbf{F}_n + \frac{\|\mathbf{P}(t) - \mathbf{P}_n\|}{\|\mathbf{P}_n - \mathbf{P}_{n-1}\|}(\mathbf{F}_n - \mathbf{F}_{n-1})$$

Cette extrapolation linéaire en position n'est cependant plus satisfaisante dès que la géométrie du maillage à proximité de l'outil varie un peu fortement. Les défauts sont particulièrement sensibles dès qu'il existe des variations dans la direction de la force émise (voir figure 4.20) et se traduisent principalement par des à-coups et des instabilités. Dès que l'on débute la découpe, le retour d'effort était donc débranché. La volonté de maintenir active la capacité de retour d'effort tout le long de l'opération nous a donc conduit à reconsidérer la manière avec laquelle celui-ci était implémenté dans le simulateur.

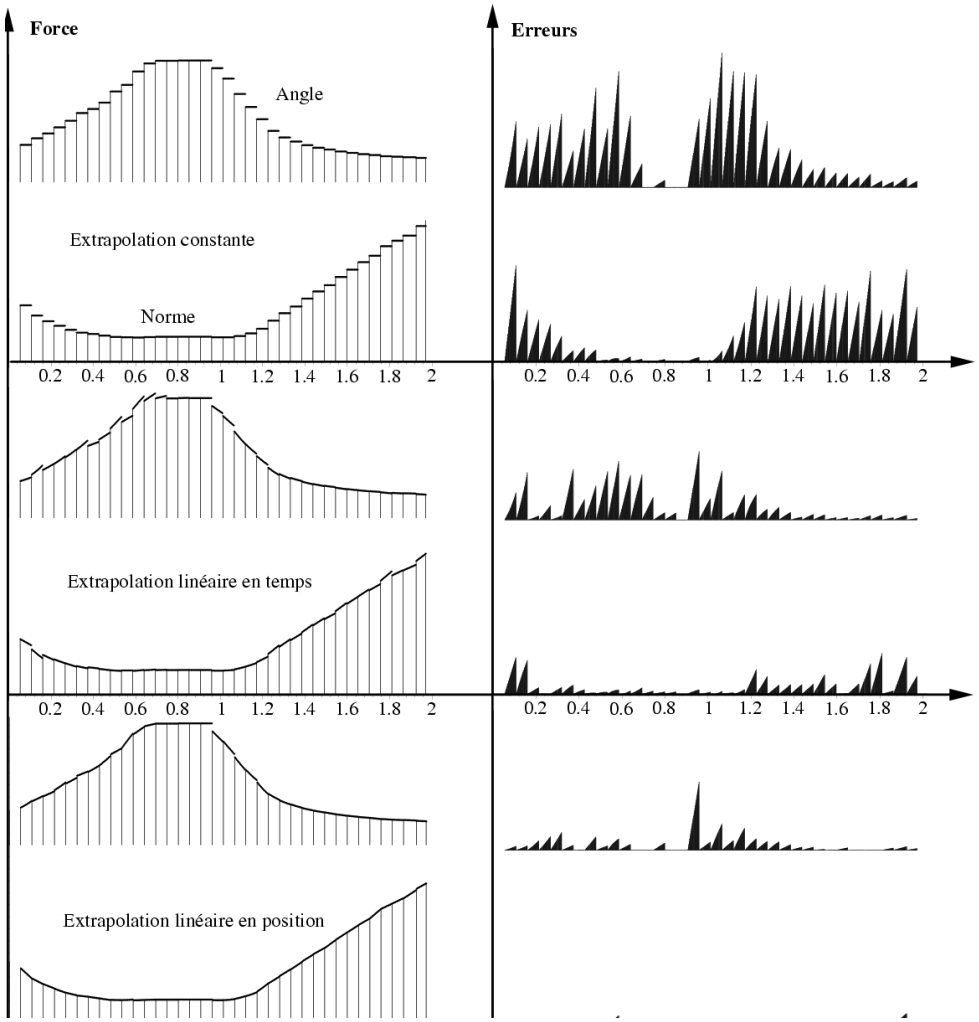


FIG. 4.20 – *Comparaison des trois méthodes d'extrapolation (constante, linéaire en temps, linéaire en position) et de leurs erreurs respectives. (source [Picinbono, 2001])*

4.5.3 Modèle local

La technique employée actuellement utilise le concept de *modèle local* que l'on trouve en plusieurs endroits de la littérature [Mendoza Serrano and Laugier, 2001; Balaniuk, 1999; Mark *et al.*, 1996]. L'idée est de transmettre au processus chargé du calcul des forces une estimation de la surface du maillage à proximité de l'outil virtuel : *le modèle local*. Le processus utilise alors ce modèle pour déterminer en temps réel de combien l'outil virtuel pénètre la surface du maillage et ainsi estimer la force qui doit être émise. À chaque pas de simulation, le modèle local est mis à jour. La nature même de ce modèle local varie selon les mises en œuvre : plan, sphère, ou même maillage. Le plupart de ces méthodes sont cependant conçues pour un outil ponctuel, ce qui simplifie la détermination, le calcul et la représentation du modèle local.

On a vu que les contacts avec l'outil virtuel peuvent être de deux types : ceux avec la pointe et ceux avec le manche. En s'inspirant des considérations faites précédemment (cf. § 4.4.4), on peut modéliser naturellement le voisinage de la pointe par un plan passant par $\mathbf{P}_{prox}$ et de normale $\mathbf{n}_{prox}$. Le manche n'étant pas ponctuel, son voisinage est plus complexe à représenter. Si on ne prend en compte que les contacts avec le manche, l'ensemble des positions non accessibles par l'outil est un cône de $I\!R^3$ dont le centre est situé sur le point fixe de l'outil. Dire que l'outil occupe une position non accessible est alors équivalent à dire que sa pointe est située à l'intérieur de ce cône. Au premier ordre, on modélise donc le voisinage du manche par un plan passant par le point fixe. La normale de ce plan est définie comme étant la moyenne des normales aux plans définis d'une part par le point fixe de l'outil, et d'autre part par les arêtes situées à proximité du manche (voir figure 4.21). Le plan est ensuite translaté selon cette normale d'une distance égale au rayon de l'outil afin de prendre en compte le volume de celui-ci.

La force que l'on envoie à l'utilisateur est calculée de manière géométrique. Elle est proportionnelle à la pénétration par la pointe de l'outil des plans qu'on vient de définir. S'il y a simultanément plusieurs contacts avec l'outil, on combine les deux vecteurs de force obtenus. Dans le cas où le maillage est en prise, la force est proportionnelle à la distance avec le plan correspondant.

On observe cependant un problème avec cette méthode : chaque fois que le modèle local est mis à jour, ie. à chaque pas de simulation, on crée une discontinuité qui se révèle être assez sensible. L'idée est alors de passer de manière progressive de l'un des modèles à l'autre en envoyant à l'utilisateur une force $\mathbf{F}(t)$ définie par :

$$\begin{aligned}
&\text{pour } t \in [t_n, t_n+\delta] \ \mathbf{F}(t) = \frac{(t-t_n)}{\delta}.\mathbf{F}_n(t) + \frac{(t_n+\delta-t)}{\delta}.\mathbf{F}_{n-1}(t),\\
&\text{pour } t \in [t_n+\delta, t_{n+1}] \ \mathbf{F}(t) = \mathbf{F}_n(t)
\end{aligned}$$

où $\mathbf{F}_{n-1}(t)$ et $\mathbf{F}_n(t)$ représentent les forces calculées en utilisant les modèles locaux définis respectivement aux itérations $n-1$ et n, et où δ est un intervalle de temps plus petit que $t_{n+1} - t_n$.

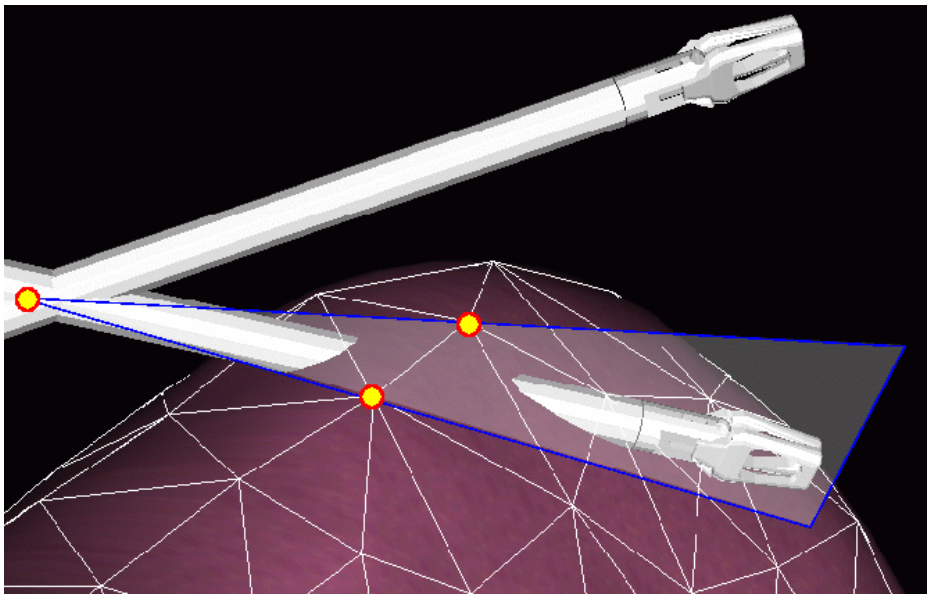


FIG. 4.21 – *La pénétration du manche de l'outil dans le maillage est équivalente à celle de sa pointe dans un plan tangent au maillage et passant par le point fixe de l'outil.*

4.6 Conclusion

Nous avons proposé une méthode permettant le traitement des collisions entre le maillage et un outil virtuel de forme linéaire. Du fait de la faible fréquence des itérations, nous avons renoncé aux méthodes par pénalité pour utiliser des méthodes géométriques qui garantissent la non pénétration du maillage par l'outil. Le retour d'effort est une composante essentielle du réalisme de la simulation. Son introduction implique que l'on soit capable de générer des forces à une fréquence suffisante (typiquement 500 Hz). Nous proposons une adaptation de la technique du modèle local au cas où l'outil n'est pas ponctuel, mais linéaire, qui permet d'atteindre sans problème cette fréquence.

Il resterait encore plusieurs améliorations à apporter. Premièrement il faudrait considérer explicitement le cas où plusieurs outils virtuels agissent sur le même maillage. En effet, les modifications du maillage engendrées par un outil peuvent créer une pénétration d'un outil déjà traité qui ne sera pas détectée. Deuxièment, et du point de vue du retour d'effort, il faudrait rapidement arriver à envoyer un signal lorsque les outils virtuels entrent en contact l'un avec l'autre. À plus long terme, il serait aussi intéressant de passer d'une détermination purement géométrique des forces renvoyées à l'utilisateur à une détermination vraiment physique.

Chapitre 5

Introduction des réseaux vasculaires

Sommaire

5.1 Vascularisation du foie

Le foie est un organe très fortement vascularisé. Le *réseau porte*, qui pénètre dans le foie au niveau du *hile*, suit les ramifications du tissu conjonctif formé par le prolongement de la *capsule de Glisson*. Ce réseau est chargé de répartir le sang provenant de l'intestin vers les *lobules hépatiques* où il est filtré. Parallèlement au réseau porte, et suivant les mêmes ramifications, on trouve le *réseau artériel hépatique*, chargé de l'oxygénation du foie, et le *réseau biliaire* qui, via les *canicules biliaires*, collecte la bile sécrétée par les *cellules hépathocytes* et la fait migrer, via le *canal cystique*, vers la vésicule biliaire et le pancréas. On emploie fréquemment le terme de *triade portale* pour désigner l'ensemble de ces trois réseaux. Le sang des réseaux porte et hépatique est évacué des lobules par les *veines centro-lobulaires* qui se réunissent pour former le *réseau sus-hépatique*. On observe donc des réseaux de deux géométries différentes : le réseau porte et le réseau sus-hépatique, en contact l'un avec l'autre à leurs extrémités, au niveau des lobules.

L'importance de la vascularisation n'est pas sans conséquences sur le déroulement d'une opération telle que l'hépatectomie. Lors de la découpe du parenchyme, il est en effet très fréquent de rencontrer des vaisseaux. De plus, la résection d'un ou plusieurs segments passe forcément par la découpe des principales veines les irriguant. Si l'une de ces veines est coupée par mégarde, l'hémorragie causée augmente sensiblement la quantité de sang perdu par le patient. De plus, la présence de ce sang va gêner la visibilité et rendre difficile la cautérisation et la poursuite de l'opération.

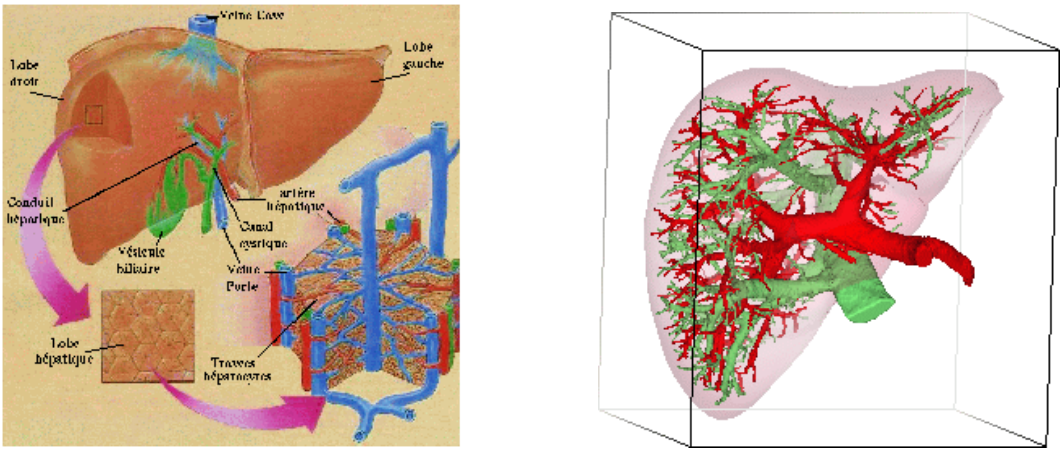


FIG. 5.1 – *À gauche : description de la structure des réseaux vasculaires hépatique. (source [Lefevre, 1994]) Pour information, un lobule hépatique a la forme d'un prisme hexagonal d'un rayon de 0,7mm et d'une hauteur d'environ 2mm. À droite : vue des arbres porte (en rouge) et sus-hépatique (en vert) obtenus par segmentation (source [Soler, 1998]).*

Il existe plusieurs techniques pour procéder à la découpe des veines sans causer d'hémorragie. Pour les veinules les plus petites, telles que les veines centro-lobulaires, on peut procéder directement à une cautérisation à l'aide du bistouri électrique. Pour

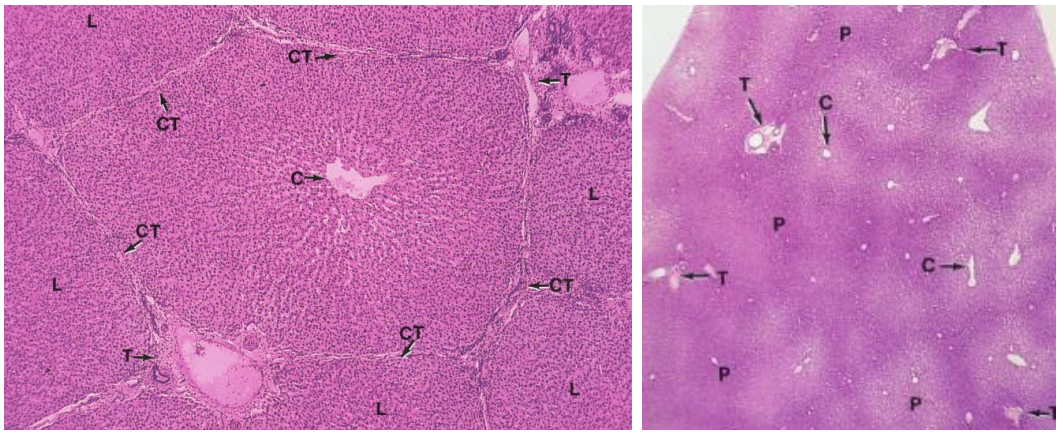


FIG. 5.2 – *Vues de lobules hépatiques de cochon (à gauche) et humain (à droite). On remarque que dans le premier cas le tissu conjonctif séparant les différents lobules est bien plus visible que chez l'homme. Légende : **C** : veine centro-lobulaire, **CT** : tissu conjonctif, **L** : lobule, **T** : triade portale. [Tor, 2003b]*

les veines plus grosses on peut commencer par les dégager, soit manuellement, soit à l'aide d'un bistouri à ultrasons, puis les clamper, c'est-à-dire mettre des agrafes de part et d'autre, et découper entre les agrafes.

5.2 Description des vaisseaux

La description suivante s'inspire grandement de l'ouvrage de biomécanique de Y.C. Fung [Fung, 1993] ainsi que de l'Atlas de Poche de A. Despopoulos [Despopoulos et Silbernagl, 1999].

5.2.1 Description générale des vaisseaux sanguins

Les vaisseaux sanguins sont des structures tubulaires fortement ramifiées présents dans l'ensemble du corps humain. Comme pour les autres tissus vivants, leur structure interne est complexe et conditionne un grand nombre de propriétés mécaniques telles que anisotropie, non-linéarité ou relaxation. Plus que pour la majorité des autres tissus, le facteur physiologique est important lorsque l'on désire en obtenir une modélisation réaliste. L'apparence et les propriétés mécaniques des vaisseaux sont en effet fortement modifiées lorsque surviennent une baisse de tension, un changement de rythme cardiaque, une coupure ou une occlusion dans une partie amont ou aval du réseau.

Il existe plusieurs motivations à la modélisation des vaisseaux. On peut s'intéresser à leurs structure physique, pour améliorer les techniques de segmentation [Verdonk, 1996] ou de détection de sténose artérielles [Krissian, 2000]. On peut aussi étudier leurs comportement mécanique dans le cadre d'un simulateur de chirurgie

tel que le nôtre, dans celui d'un simulateur de microchirurgie [Brown *et al.*, 2001a; Brown *et al.*, 2002] ou pour la conception de nouveaux outils ou de nouvelles prothèses [Windecker *et al.*, 2001]. On peut aussi se préoccuper principalement de la modélisation de l'écoulement sanguin, pour évaluer la propagation de substances anesthésiantes [Olufsen, 1998]. Dans tous les cas, si on veut obtenir une modélisation robuste et réaliste, on devra s'intéresser de près aux modifications entraînées par des variations de grandeurs physiologiques telles que le rythme ou la tension cardiaque, le stress, etc.

On peut distinguer trois types de vaisseaux bien distincts :

- les artères, dont l'ensemble constitue le réseau artériel. Elles sont chargées de la distribution du sang depuis le cœur vers tous les organes du corps.
- les veines, qui constituent le réseau veineux, et dont le but est de collecter le sang depuis ces organes et de le rediriger vers le cœur. Elles jouent aussi un rôle non négligeable de réservoir sanguin.
- les capillaires, qui sont les vaisseaux reliant les réseaux artériel et veineux. Ils irriguent directement les cellules du corps. De par leur petite taille ($< 9\mu m$), les capillaires sont très difficilement discernables en imagerie. On ne s'y intéressera pas dans la suite.

Artères

Il est commode d'effectuer une classification par la taille des différentes artères. On peut ainsi distinguer :

- l'aorte et l'artère pulmonaire, qui sortent du cœur et dont le diamètre est égal à 2-3 centimètres.
- les grosses artères, de diamètre compris entre 0,8 et 2 centimètres, qui répartissent le sang vers la périphérie,
- les branches artérielles dont le diamètre varie entre 0,2 et 0,8 centimètre, qui acheminent le sang jusqu'aux organes,
- les artérioles, dont le diamètre varie entre 0,02 et 0,2 centimètres, qui répartissent le sang à l'intérieur des organes.

La paroi des artères est de section cylindrique, relativement rigide et possède une épaisseur qui peut se diviser en trois composantes :

L'intima Partie la plus interne, l'intima est composée de cellules endothéliales fixées sur une membrane conjonctive appelée lame basale (*basal lamina*). Ce sont ces cellules qui forment la surface interne du vaisseau. L'intima des artères les plus grosses peut de plus contenir une couche dite sub-endotheliale, composée, entre autres, de quelques cellules musculaires.

La media Partie intermédiaire de la paroi, la media est formée de tissu conjonctif (fibres d'élastine et de collagène) ainsi que d'un grand nombre de cellules musculaires. Ces dernières sont entrecroisées selon un motif hélicoïdal dont l'inclinaison diminue lorsqu'on s'approche de la périphérie.

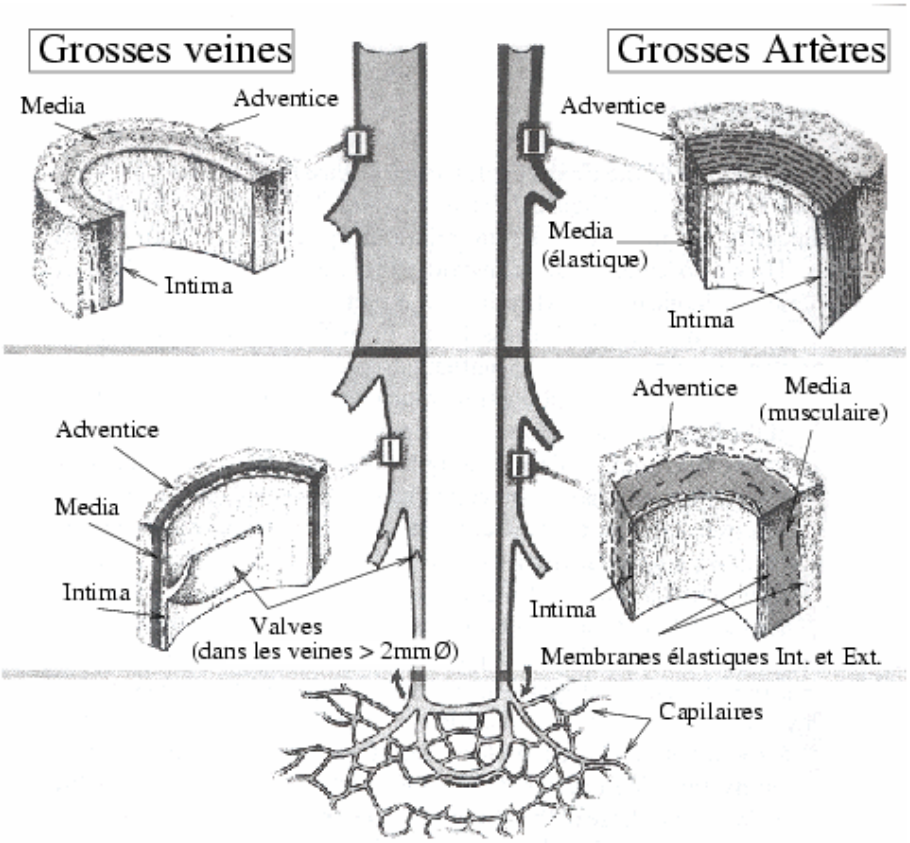


FIG. 5.3 – *Structure pariétale des principales veines et artères, d'après [Rhodin, 1980]*

.

La présence de ces cellules musculaires dans la media est importante. D'une part elles participent à structurer l'artère et assurent ainsi sa rigidité. D'autre part elles permettent l'existence de phénomènes de vasomotricité : vasoconstriction et vasodilatation. La raison d'être de ces phénomènes est de permettre la régulation du débit et de la pression dans le circuit artériel. Ils peuvent avoir lieu soit sur un stimulus nerveux externe au vaisseau soit sur un stimulus interne comme lors d'une variation de la pression sanguine.

L'adventice La partie externe de l'artère est nommée adventice. Elle est composée d'un tissu conjonctif qui sert de support au *vasa vasorum*, ensemble de vaisseaux servant à l'irrigation des cellules de l'artère, et au *nervi vasorum* cellules nerveuses myelinisées et non-myelinisées servant à son innervation et donc au contrôle des effets de vasomotricité.

En général, plus on s'éloigne de l'aorte, plus l'épaisseur relative de la paroi est grande. L'importance relative de trois parties constituant la paroi varie elle aussi avec le type et la taille de l'artère. Les artères cérébrales ont la particularité de ne pas posséder d'adventice. Elles possèdent une vasomotricité particulière qui leur permet une grande vasodilatation ou, à l'inverse, un vasospasme.

Veines

De manière parallèle, on classifie les veines par leur taille. On distingue donc :

- les deux veines caves, dont le diamètre est d'environ 3 centimètres.
- les grosses veines, d'un diamètre de 0,7 à 2 centimètre,
- les branches veineuses dont le diamètre peut varier entre 0,15 et 0,7 centimètre,
- les veinules, de diamètre variant entre 0,2 et 1,5 millimètre,

Les veines ont une structure pariétale différente de celle des artères. L'intima de la plupart des principales veines comporte des petites valvules en forme de demi-lune qui se comportent comme des valves et empêchent le sang de refluer. Remarquons cependant que le système de la veine porte ne comporte pas de valve. La media et l'adventice sont très fines et ne comportent que peu de fibres musculaires. Cette absence de fibres musculaire et la relative finesse de la paroi joue sur les propriétés mécaniques des veines. Elles ne sont en effet pas aussi rigides que les artères et sont beaucoup plus sensibles aux effets des pressions pariétales interne (pression sanguine) et externe (pression des organes de voisinage). Sauf pour les plus grosses, leur section n'est donc pas cylindrique.

5.2.2 Rhéologie

Généralités

Un très grand nombre de travaux ont été réalisés pour déterminer le comportement mécanique des veines et des artères.

L'un des points les plus importants est que les vaisseaux ne vérifient pas la loi de Hook sur l'élasticité linéaire [Fung, 1993]. Leur comportement présente par exemple des hystéresis : à étirement constant on observe une relaxation de la tension, à charge constante l'étirement augmente avec le temps. Malgré cela, l'hypothèse d'élasticité reste communément employée.

Les veines se comportent globalement de la même manière que les artères, leur structure interne n'étant que peu différente.

Étirement longitudinal

La structure tubulaire des vaisseaux permet de simplifier grandement l'étude de leur comportement mécanique en modélisant leur paroi comme une simple surface. Les grandeurs mécaniques sont alors considérées comme étant constantes sur toute l'épaisseur de la paroi. Il existe bien sûr des problèmes ne permettant pas cette simplification, par exemple l'étude du comportement mécanique lors de pliures ou celle des effets de la sténose ou de l'anévrisme.

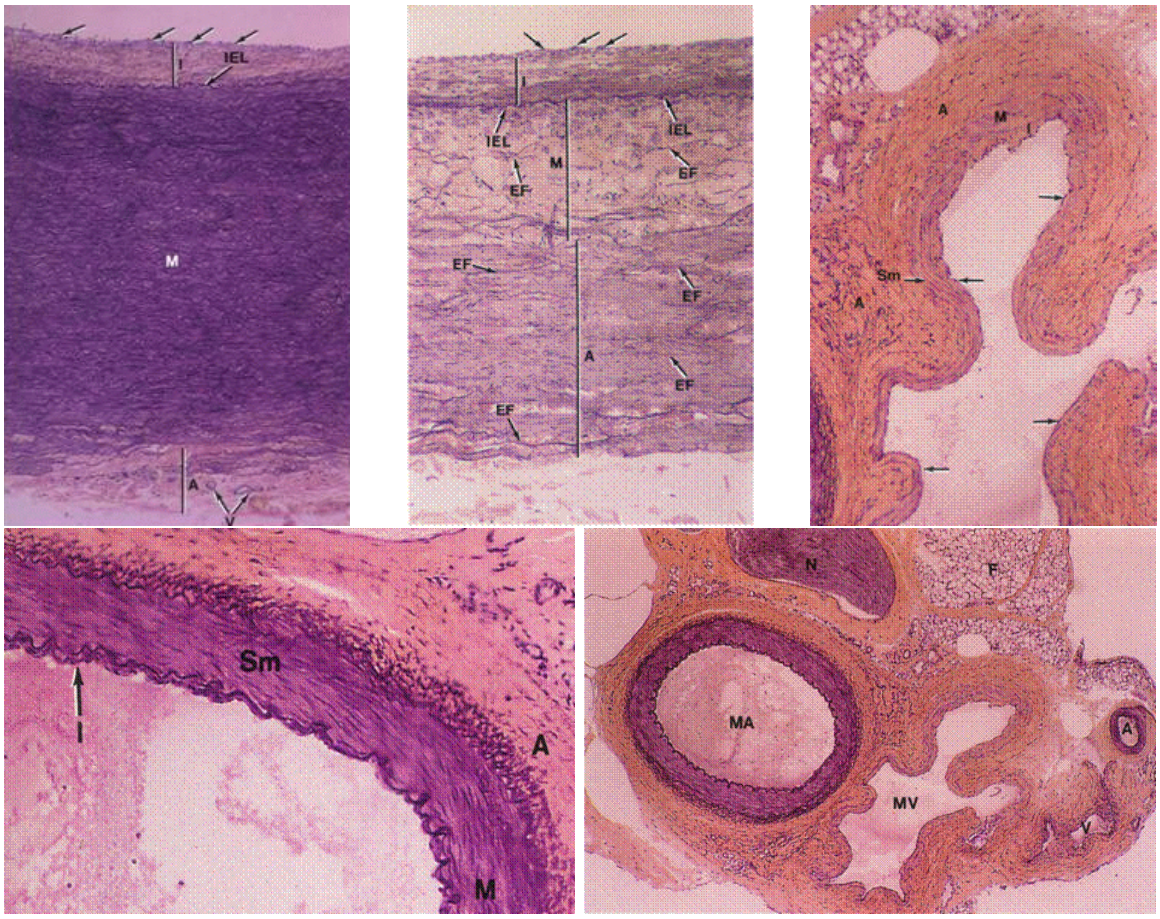


FIG. 5.4 – *Vue en coupe de la structure pariétale, dans l'ordre : d'une grosse artère, de la veine cave, d'une branche artérielle et d'une branche veineuse. En bas à droite on peut comparer le profil circulaire de l'artère à celui beaucoup moins régulier de la veine. Légende : **A** : adventice, **M** ': media, **Sm** : cellules musculaires, **I** : intima, **IEL** : lame basale, **EF** : cellules élastiques. (source [Tor, 2003a]).*

La première remarque que l'on peut faire sur le comportement mécanique des vaisseaux, et plus généralement sur celui des tissus mous, est qu'ils ne vérifient pas la loi de Hooke de proportionnalité entre la contrainte et l'étirement. En fait, ces matériaux ne sont même pas hyperélastiques car on observe l'existence d'hystérésis. Fung [Fung, 1993] fait l'hypothèse que la réponse K en contrainte d'un tissu mou à un étirement spontané $\lambda = \frac{l}{l_0}$ peut se décomposer en un produit de deux fonctions, l'une ne dépendant que de λ et l'autre ne dépendant que du temps.

$$T(t,\lambda) = G(t).T^{(e)}(\lambda), \tag{5.1}$$

La fonction G est une fonction continue décroissante, appelée *fonction de relaxation*, valant 1 à l'origine et tendant vers une limite $G(\infty)$. La fonction $T^{(e)}$ est appelée *réponse élastique*. Il suppose de plus que l'on peut appliquer le principe de superposition, c'est-à-dire que si la réponse à un petit étirement $\delta\lambda(\tau)$ (obtenue par dérivation de l'équation 5.1) vaut

$$G(t-\tau).\frac{\partial T^{(e)}[\lambda(\tau)]}{\partial \lambda}.\delta\lambda(\tau) \tag{5.2}$$

alors la contrainte à l'instant t vaut :

$$T(t) = \int_{-\infty}^{t} G(t-\tau).\frac{\partial T^{(e)}[\lambda(\tau)]}{\partial \lambda}.\frac{\partial \lambda(\tau)}{\partial \tau}.d\tau \tag{5.3}$$

Le terme $\frac{\partial T^{(e)}[\lambda(\tau)]}{\partial \lambda}$ représentant le module d'Young du matériau pour l'étirement $\lambda(\tau)$ et $\frac{\partial \lambda(\tau)}{\partial \tau}$ caractérisant la vitesse d'étirement du matériaux. Dans le cas d'artères et de veines, on observe alors deux types de comportement en fonction de l'intensité de la contrainte exercée. Si cette contrainte est inférieure à une valeur de 20 kPa, alors contrainte et étirement sont reliés par une relation du type :

$$T = \gamma.\lambda^k. \tag{5.4}$$

Si cette contrainte est supérieure à 20 kPa, alors le module d'Young $\frac{dT}{d\lambda}$ devient linéaire par rapport à T et on a :

$$\frac{dT}{d\lambda} = \alpha.(T+\beta) \tag{5.5}$$

d'où une relation de la forme :

$$T = (T^* + \beta).e^{\alpha.(\lambda-\lambda^*)} - \beta \tag{5.6}$$

Les valeurs des constantes α, β, γ, k, λ^* et T^* varient grandement en fonction de la localisation du vaisseau et de sa nature (artère ou veine).

Rappelons qu'une contrainte de 20 kPa correspond à une force de 6 N pour une veine d'un diamètre d'environ 2 cm qui est le diamètre maximum typique de la veine porte et de 0,06 N pour une veine de diamètre 2 mm, qui est le diamètre des plus petites veines accessibles par les méthodes de segmentation utilisées [Soler, 1998].

L'une des études qui peut particulièrement nous intéresser est la variation de la contrainte lors d'un étirement constant. On a vu que l'équation 5.1 pouvait représenter

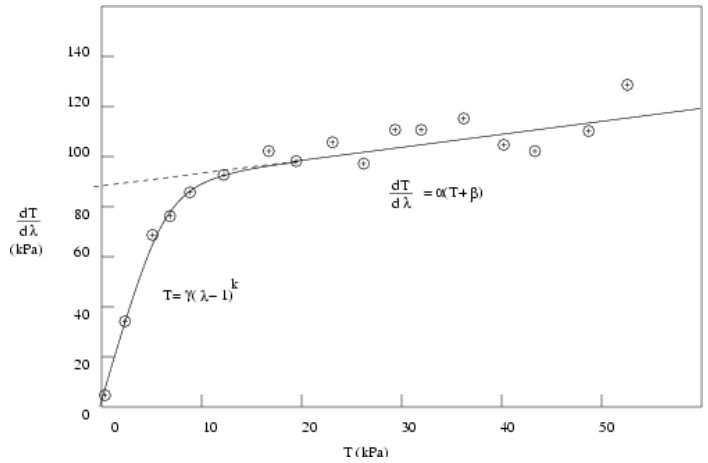


FIG. 5.5 – *Graphique montrant la valeur du module d'Young $\frac{dT}{d\lambda}$ en fonction de la contrainte dans le cas d'une artère de chien (source [Fung, 1993]).*

l'évolution dans le temps de la réponse à un étirement donné λ. La forme classique utilisée pour la fonction de relaxation est :

$$G(t) = \frac{\sum C_i.e^{-\nu_i.t}}{\sum C_i} \tag{5.7}$$

Pour intégrer certains résultats sur l'hystérésis, en particulier l'absence de fréquence de résonance, on utilise parfois un spectre continu de ν_i.

Effet de la pression sur le diamètre

La plupart des auteurs font l'hypothèse, qu'ils savent fausse, du caractère hyperélastique de la paroi des vaisseaux [Fung, 1993; Olufsen, 1998]. Les expériences cherchent alors à déterminer la valeur du module d'Young dans leur direction radiale. Les parois sont maintenues à leur longueur naturelle et sont étirées dans leur direction radiale. M. Olufsen observe, entre le module d'Young E du vaisseau, son rayon r_0, et l'épaisseur h de sa paroi, une relation de la forme :

$$\frac{Eh}{r_0} = k_1 e^{k_2 r_0} + k_3 \tag{5.8}$$

qui est valable pour les artères de diamètre supérieur à 0.8 cm mais qu'elle affirme pouvoir étendre aux artères de plus faible diamètre.

En supposant que l'épaisseur h de la paroi est suffisamment faible pour qu'on puisse la négliger dans l'analyse mécanique, on peut relier la contrainte tangentielle et le module d'Young par la relation :

$$T_\theta = \frac{Eh}{1-\nu^2}\frac{\Delta r}{r_0} \tag{5.9}$$

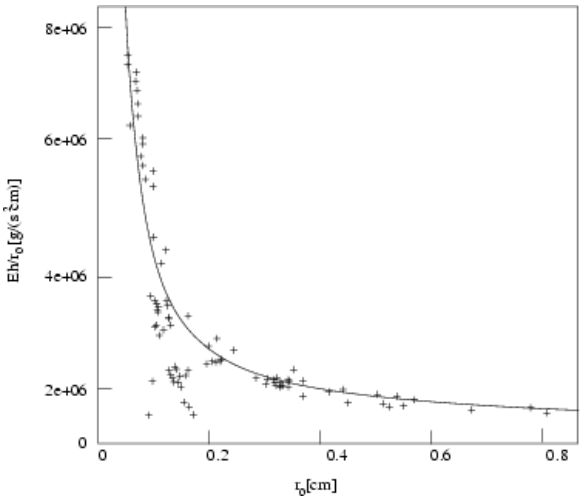


FIG. 5.6 – *Liaison entre le module d'Young E, le rayon r_0 et l'épaisseur de la paroi h dans le cas d'artères humaines (d'après [Olufsen, 1998])*

où ν représente la valeur du coefficient de Poisson pour le vaisseau que l'on prend généralement égal à $\frac{1}{2}$. Pour une telle valeur, l'équilibre du système fait donc dépendre la variation de rayon du vaisseau avec la variation de pression de celui-ci par :

$$\Delta r = \frac{4}{3}\frac{r_0 \Delta P}{Eh} \tag{5.10}$$

5.3 Modélisation mécanique

5.3.1 Différentes possibilités

On peut imaginer trois possibilités pour insérer les vaisseaux dans le modèle mécanique du foie. La première serait d'intégrer les vaisseaux directement dans le maillage du foie. On donne aux tétraèdres correspondants des caractéristiques mécaniques spécifiques et on résout l'ensemble du système normalement. C'est la méthode qui a été employée par la société ESI dans le cadre de l'action CAESARE. La difficulté de cette méthode est qu'elle multiplie énormément le nombre de tétraèdres et n'est donc pas utilisable dans le cadre d'une simulation temps réel. On pourrait cependant imaginer son utilisation dans le cadre d'un modèle précalculé.

Une autre méthode qui a été employée dans des versions précédentes du simulateur [Picinbono, 2001; Picinbono *et al.*, 2001a] est de ne pas modifier le maillage du foie mais de modifier le comportement des tétraèdres traversés par les veines principales, par exemple en y introduisant une forte anisotropie. Le problème de ce modèle est qu'il n'intègre pas, en tant que tel, la possibilité d'accéder aux veines lors de la découpe.

L'idée que l'on a mise en œuvre est plus souple et consiste à faire cohabiter et interagir deux modèles différents, d'une part le parenchyme et d'autre part les vaisseaux.

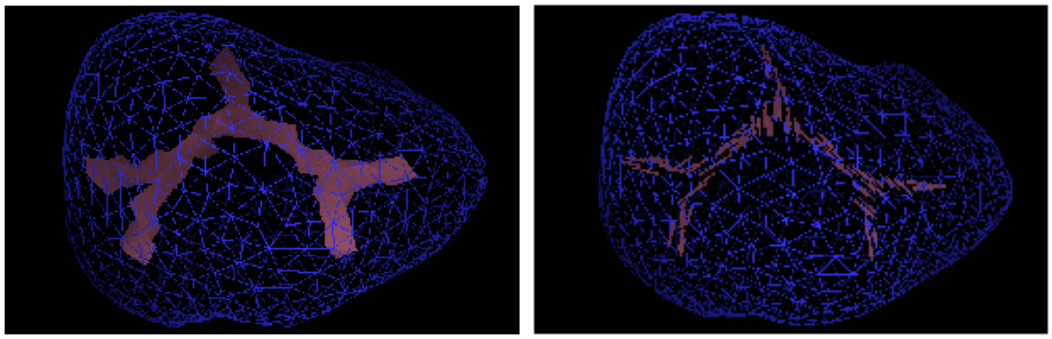


FIG. 5.7 – *Les tétraèdres contenant les branches principales de l'arbre porte sont représentés en rouge sur l'image de gauche. À droite on a représenté les directions d'anisotropie. (source [Picinbono, 2001])*

Il existe, là encore, plusieurs possibilités de concevoir la modélisation des vaisseaux, et le choix va dépendre de la complexité que l'on souhaite donner au modèle, de la précision désirée et de l'usage attendu. Si on cherche à simuler l'insertion de stents par exemple, on aura intérêt à utiliser un modèle volumique. Si on cherche à simuler des opérations de ligature en microchirurgie [Brown *et al.*, 2002], on se dirigera plutôt vers une modélisation surfacique. Dans notre cas, les seules opérations que l'on désire effectuer sur les vaisseaux sont la manipulation, la pose d'agrafes et la découpe. On peut alors se contenter d'un modèle linéique.

5.3.2 Modèle linéique

La géométrie des vaisseaux est extraite à partir d'une image scanner de l'abdomen du patient à l'aide d'une méthode de segmentation de type seuillage dont le résultat est affiné avec des techniques d'amincissement et d'analyse de graphe [Soler, 1998; Selle *et al.*, 2002]. Les résultats sont traités de façon à réduire les erreurs dues, d'une part à la distance entre les différentes coupes du scanner, et d'autre part à la difficulté liée à la séparation des tissus par seuillage. Le squelette du réseau est construit puis épuré et la topologie des arbres vasculaires est reconstruite. Pour la veine porte par exemple, on obtient un arbre composé d'une soixantaine de branches et d'environ deux mille sommets. Pour chacun de ces sommets, on dispose en plus d'une donnée scalaire correspondant au rayon de la veine en ce sommet.

Une branche du réseau veineux est donc modélisée par un ensemble de *sommets* reliés deux à deux par des *arêtes*, et affecté d'une valeur scalaire appelée *rayon*. Plusieurs branches peuvent être reliées les unes aux autres par leurs extrémités afin de représenter les embranchements du réseau. Cette représentation est très simpliste et d'autres modélisations seraient souhaitables si on voulait donner aux veines des comportements plus réalistes, en particulier si on voulait limiter les risques d'auto collision ou de collision avec le parenchyme [France *et al.*, 2002].

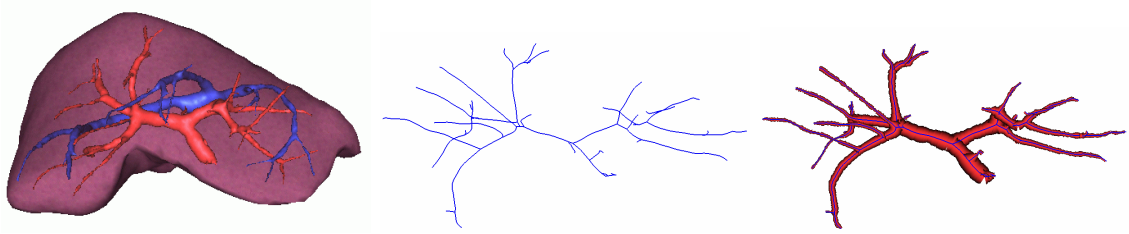


FIG. 5.8 – *A droite, obtention de la géométrie des vaisseaux après la segmentation d'une image scanner volumique. En rouge la veine porte, en bleu la veine cave. Au centre et à droite, modèle linéique du vaisseaux et son habillage cylindrique.*

5.3.3 Structure de données

La classe `C++` de vaisseau hérite de la classe de contour déjà présente dans `yav++` et qui est utilisée pour effectuer des recalages ou visualiser l'intersection de maillages avec des plans. L'intérêt d'hériter de cette classe était de pouvoir profiter de l'ensemble des fonctionnalités de cette dernière: structure de données, calcul des déformations, etc. À l'usage, cet héritage se révèle plutôt handicapant, la problématique ayant conduit à la création de la classe contour se révélant relativement différente de la nôtre. On envisage donc de développer une classe particulière.

5.3.4 Insertion dans le maillage

Au début de la simulation, les vaisseaux sont plongés dans le parenchyme et sont solidaires avec celui-ci. On fixe donc la position des sommets des veines relativement à celle des sommets du maillage. Soit P_V est la position de l'un des sommets V des veines, à chaque instant t, on a :

$$P_V(t) = \sum_{S \in T} \alpha_{S,V} P_S(t)$$

où les $P_S(t)$ sont les positions à l'instant t des sommets du tétraèdres T dans lequel était plongé V au début de l'opération et les $\alpha_{S,V}$ sont les coefficients barycentriques de P_v à l'instant $t = 0$.

Réciproquement, on rajoute au terme de l'énergie élastique du maillage un terme correspondant aux déformations des veines :

$$W_{total} = W_{maillage} + W_{veines}$$

En gardant les notations précédentes, la force $F_S(t)$ s'exerçant sur un sommet S du maillage à l'instant t vaut :

$$\begin{aligned} F_S(t) &= \frac{\partial W_{total}}{\partial P_S}(t) \\ &= \frac{\partial W_{maillage}}{\partial P_S}(t) + \frac{\partial W_{veines}}{\partial P_S}(t) \end{aligned}$$

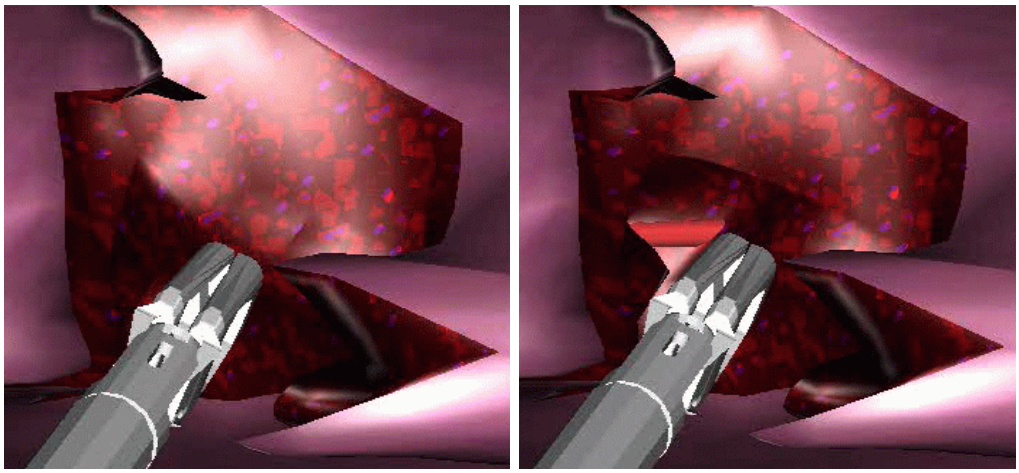


FIG. 5.9 – *La suppression d'un tétraèdre contenant un vaisseau entraîne l'apparition de celui-ci.*

$$\begin{aligned} &= F_{maillage} + \sum_V \alpha_{S,V} \frac{\partial W_{veines}}{\partial P_V}(t) \\ &= F_{maillage} + \sum_V \alpha_{S,V} F_V(t) \end{aligned}$$

où F_V peut être vue comme la force interne au vaisseau qui s'exerce sur le sommet V. On peut interpréter cette relation comme un échange force/position : le maillage fixe la position du vaisseau et en contrepartie celui-ci lui transmet ses forces internes. L'énergie W_{veines} que l'on veut utilise doit correspondre à la rhéologie des veines. On utilise un modèle masse-ressort. Celui-ci a été choisi originellement du fait de sa grande simplicité et son usage a été maintenu car il est assez satisfaisant. L'énergie correspondante est donc du type :

$$W_{veines}(t) = \frac{1}{2} \sum_{E \in \mathcal{E}} k_E . \Delta E(t)^2$$

où $\mathcal{E}$ représente l'ensemble des arêtes du réseau et ΔE l'étirement d'une arête E. Il est toutefois possible de complexifier ce modèle : on peut par exemple rajouter des ressorts angulaires.

Si le tétraèdre dans lequel est inclus un sommet V donné est raffiné, il faut déterminer à quel tétraèdre des tétraèdres résultants appartient V et recalculer les $\alpha_{S,V}$ correspondants. Pour cela on utilise la méthode `setProperties` décrite partie 2.7.2, chaque nouveau tétraèdre devant vérifier s'il contient les sommets du vaisseau contenus dans le tétraèdre d'origine.

Si le tétraèdre dans lequel est inclus un sommet donné est supprimé, ce sommet est *libéré*. Les sommets libérés peuvent alors se déplacer indépendamment des sommets du maillage du parenchyme. Sur chacun d'eux s'exerce une force

$$F_V(t) = \frac{\partial W_{veines}}{\partial P_V}(t).$$

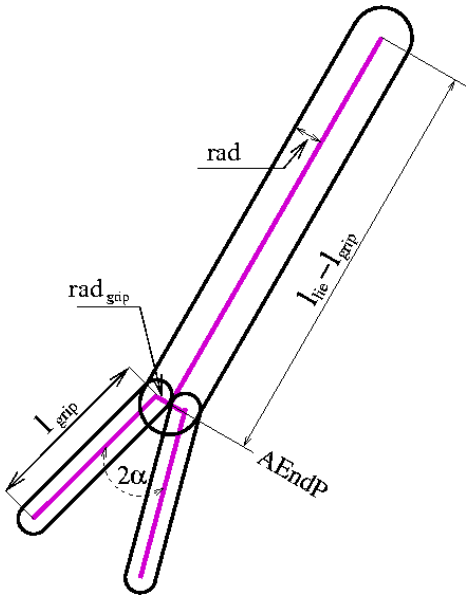


FIG. 5.10 – *Modèle d'outil virtuel utilisé lors du traitement des collisions avec les vaisseaux.*

Pour accélérer le rendu graphique, seules les arêtes en contact avec des sommets libérés sont affichées.

5.4 Interaction avec les outils

5.4.1 Description de l'outil

Le modèle d'outil virtuel que l'on utilise pour l'interaction avec les vaisseaux est, comme on l'a vu au cours de la partie 4.3, le dilaté de quatre segments (cf. figure 5.10) : un segment pour le manche, un segment pour chacune des pinces, et un segment placé à l'articulation.

5.4.2 Traitement des collisions

Comme pour le maillage, le traitement des collisions avec les vaisseaux se déroule en trois étapes, éventuellement précédées d'une phase consistant à déterminer les arêtes proches de l'outil et susceptibles d'être concernées par le processus de traitement des collisions. Premièrement on détecte les franchissements du squelette par les arêtes qui ont eu lieu au cours du pas de temps précédent. Ensuite, on modifie les positions des sommets afin d'empêcher ce franchissement. Enfin, on repousse l'ensemble des arêtes hors du volume de l'outil.

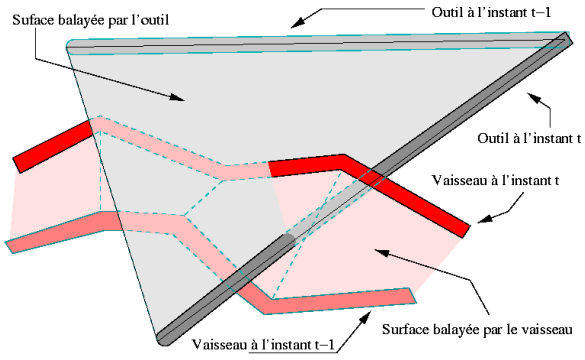


FIG. 5.11 – *Détermination des arêtes proches de l'outil.*

Détermination des arêtes proches

Le but cette étape est de sélectionner efficacement un sous-ensemble d'arêtes susceptible d'entrer en collision avec l'outil et sur lesquels pourront être effectués les calculs plus complexes spécifiques à la collision et décrits plus loin. Remarquons que si le nombre total d'arêtes libres est assez faible, on peut se dispenser de cette étape préliminaire. Les calculs seront alors effectués sur la totalité des arêtes. La géométrie du parenchyme nous a permis de supposer qu'une collision se traduit toujours par un pénétration du maillage par l'outil à la fin du pas de temps courant (cf. § 4.4.1). Cette hypothèse n'est plus valable pour les vaisseaux, qui sont de forme longiligne et il est nécessaire de prendre en compte leur mouvement.

Soit E une arête, soient A, B, $A+\Delta A$ et $B+\Delta B$ la position de ses deux sommets respectivement aux pas de temps $t-1$ et t. Le quadrilatère $(A,B,B+\Delta B,A+\Delta A)$ correspond à la surface balayée par cette arête entre ces deux pas de temps. Une détection de collision de type LCN (cf. § 4.4.1) entre l'union de ces surfaces d'une part, et une caméra dont le champ de visualisation correspond au volume balayé par l'outil d'autre part, permet de repérer quelles sont les arêtes susceptibles d'entrer en collision avec ce dernier (voir figure 5.11).

Lorsque le nombre d'arêtes libérées est faible, on pourrait se dispenser de cette première étape et effectuer les calculs suivants sur la totalité des arêtes libérées.

Détection des collisions avec le squelette

Soient E une arête du vaisseau et S l'un des segments de l'outil. Soient A, B, C et D les positions de leurs extrémités respectives au pas de simulation précédent, $A+\Delta A$, $B+\Delta B$, $C+\Delta C$ et $D+\Delta D$ celles au pas de temps actuel. Il y a franchissement du segment S par l'arête E si l'on peut trouver t, β et δ dans $[0,1]\times[0,1]\times[0,l]$ vérifiant :

$$A+t.\Delta A+\beta.(B+t.\Delta B-A-t.\Delta A)=C+t.\Delta C+\delta.(D+t.\Delta D-C-t.\Delta C)$$

En prenant le produit scalaire de cette égalité avec le vecteur

$$(B+t.\Delta B-A-t.\Delta A)\wedge(D+t.\Delta D-C-t.\Delta C)$$

on peut supprimer les termes en β et en δ :

$$\Big(A - C + t.(\Delta A - \Delta C)\Big).\Big((B + t.\Delta B - A - t.\Delta A) \wedge (D + t.\Delta D - C - t.\Delta C)\Big) = 0$$

On se ramène alors à une équation du second degré en t :

$$\begin{aligned} 0 = \quad & \Big(A - C\Big).\Big((B - A) \wedge (D - C)\Big) \\ & + t.\Big(\Big(\Delta A - \Delta C\Big).\Big((B - A) \wedge (D - C)\Big) \\ & \qquad + \Big(A - C\Big).\Big((B - A) \wedge (\Delta D - \Delta C) + (\Delta B - \Delta A) \wedge (D - C)\Big)\Big) \\ & + t^2.\Big(\Big(\Delta A - \Delta C\Big).\Big((\Delta B - \Delta A) \wedge (\Delta D - \Delta C)\Big)\Big) \end{aligned}$$

On traite cette équation d'une manière identique à ce qui a été fait en 4.4.3. Si exactement l'une des deux racines correspond à un franchissement de l'axe de l'outil, alors on dit que l'arête a *franchi l'un des segments du squelette de l'outil*.

Traitement des collisions avec le squelette

On traite le problème arête par arête. Dans les cas où plusieurs franchissements sont détectés, on ne prend en compte que celui qui a eu lieu en premier. Lorsqu'un franchissement a été détecté, il faut bloquer le mouvement relatif de l'arête par rapport au segment.

Plaçons nous à l'instant t_f où l'arête E a franchi le segment S. Soient A_{t_f}, B_{t_f}, C_{t_f} et D_{t_f} les positions des extrémités respectives de l'arête E et du segment S à cet instant t_f. La première chose à faire est de repousser l'arête E de façon à ce qu'elle n'intersecte plus le segment S. Cela est fait dans une direction à la fois perpendiculaire à l'arête et au segment :

$$d = \lambda.(A_{t_f} - B_{t_f}) \wedge (C_{t_f} - D_{t_f})$$

À l'instant $t'_f = t_f - \delta t$, le vecteur reliant le point d'intersection situé sur le segment de celui situé sur l'arête vaut :

$$\begin{aligned} \varepsilon.\delta t = \quad & A + t'_f.\Delta A + \beta.(B + t'_f.\Delta B - A - t'_f.\Delta A) \\ - \quad & C + t'_f.\Delta C + \delta.(D + t'_f.\Delta D - C - t'_f.\Delta C) \end{aligned}$$

On a donc :

$$\varepsilon = \Delta C - \Delta A + \alpha(\Delta A - \Delta B) + \beta(\Delta D - \Delta C)$$

Pour que le déplacement ne fasse pas franchir le segment, il faut choisir le signe de λ de telle façon que le produit $d.\varepsilon$ soit positif. Pour qu'il n'entraîne pas le franchissement d'un autre segment, il faut de plus que λ soit suffisamment petit. On prend $|\lambda| = 0.1.rad_{grip}$.

Le mouvement du segment S de l'outil, entre les instants t_0 et t_1, peut être vu comme une transformation $\mathcal{R}_S$ associant au segment $[C,D]$ en le segment $[C+\Delta C,D+\Delta D]$ et maintenant identique le vecteur $C-D\wedge(C+\Delta C-D-\Delta D)$. Entre les instants t_f et t_1, le segment subit donc une transformation égale à $t_f * I + (1-t_f) * \mathcal{R}$. On fait donc subir à l'arête cette transformation.

Lorsqu'il n'y a pas deux arêtes successives qui franchissent le squelette de l'outil au cours du même pas de temps, la méthode proposée est efficace. Dans le cas contraire elle peut cependant conduire à des franchissements non désirés. Ces cas peuvent arriver d'une part lorsque la géométrie du vaisseau est très irrégulière, et d'autre part lorsque l'on effectue un mouvement de rotation rapide de l'outil autour de son axe alors qu'une arête est présente entre les mâchoires de la pince.

Déplacement du vaisseau hors de l'outil volumique

La dernière étape du traitement des collisions consiste à prendre en compte le volume des objets considérés. On cherche donc à définir un déplacement éloignant de manière optimum le vaisseau du squelette de l'outil. Ce déplacement doit bien sûr faire en sorte que le vaisseaux ne franchissent pas le squelette de l'outil.

Soit S l'un des segments de l'outil. Soit E une arête du vaisseau et A et B les positions de ses sommets. On note $d_{E,S}$ le vecteur distance reliant l'arête au segment. L'arête et le segment étant des objets convexes, ce vecteur est bien défini. Pour sortir l'arête E du volume de l'outil, les extrémités de cette arête doivent subir une transformation de vecteur :

$$\varepsilon_{A,S,E} = \max(0, rad_S + rad_A - \parallel d_{E,S} \parallel) * \frac{d_{E,S}}{\parallel d_{E,S} \parallel}$$

où A est l'un des deux sommets de l'arête E et où rad_S et rad_A sont respectivement les rayons du segment S et du vaisseau au sommet A.

On a alors, pour chacun des sommets du vaisseau, de 8 vecteurs de distance $d_{E,S}$ et de 8 vecteurs de déplacement $\varepsilon_{E,S}$, un grand nombre de ces derniers étant nuls. On utilise alors une heuristique qui consiste simplement à déplacer chacun des sommets du vaisseau d'un vecteur égal à la somme de ses vecteurs ε :

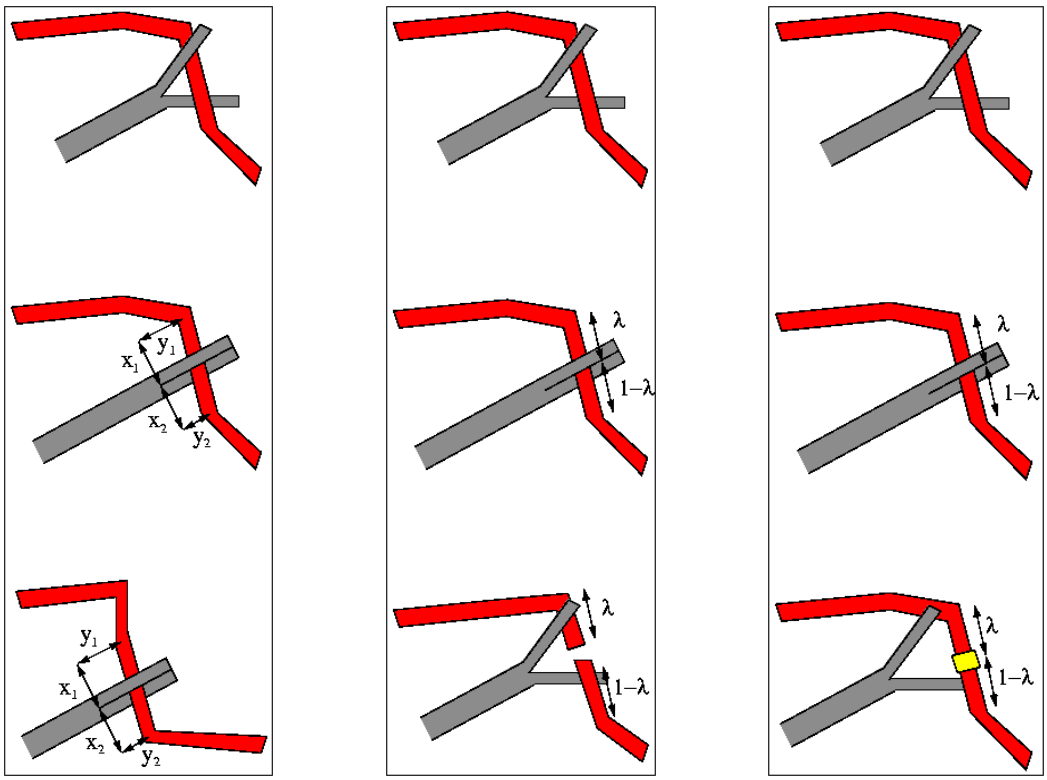
$$\varepsilon_A = \sum_{S,E} \varepsilon_{A,E,S}$$

On s'assure alors que ce déplacement ne risque de causer le franchissement d'aucun segment. Cela est fait en vérifiant que pour chacun des 8 vecteurs $d_{E,S}$ concernant le sommet A on a bien :

$$d_{E,S}.\varepsilon_A < - \parallel d_{E,S} \parallel^2$$

Ce cas arrive généralement lorsque l'on considère des arêtes situées entre les mâchoires de l'outil. On prend alors, pour le sommet, un déplacement égal à :

$$\min_{d_{E,S}.\varepsilon_A < \|d_{E,S}\|^2} \left(\frac{\parallel d_{E,S} \parallel^2}{d_{E,S}.\varepsilon_A} \right) .\varepsilon_A$$



(a) (b) (c)

FIG. 5.12 – *(a) Pour agripper on fixe les positions des sommets par rapport à l'outil. (b) Pour découper, on scinde l'arête en prise. (c) Pour clamper, on positionne une agrafe sur cette arête.*

5.4.3 Prise, découpe et clampage

On dit qu'il y a *prise du vaisseau* lorsque, lors de la fermeture de la pince de l'outil, il existe une arête du vaisseau placée entre les deux mâchoires. On mémorise alors la position relative des deux sommets de cette arête par rapport à l'outil. À chaque itération, et tant que la pince reste fermée, ces sommets seront repositionnés à la même position relative.

Lorsque l'outil est en mode *découpe* et qu'une prise du vaisseau est réalisée, celui-ci est découpé. Cela est réalisé en rajoutant un sommet au milieu de l'arête captive et en scindant ce sommet (voir figure 5.12). Si l'outil est en mode *clampage*, on place une agrafe à l'endroit de la prise.

5.5 Conclusion

Nous avons proposé une méthode permettant l'introduction des vaisseaux dans le simulateur. Ceux-ci sont représentés à l'aide d'un modèle à une dimension et ce n'est qu'à l'affichage que l'aspect volumique apparaît. Ce modèle est plongé dans le maillage volumique du parenchyme à l'aide d'une interaction force/position. Lorsque le maillage est découpé, les vaisseaux sont libérés. On propose une méthode permettant alors leur manipulation. Il serait intéressant d'améliorer le modèle mécanique utilisé pour la déformation des vaisseaux afin d'introduire certaines grandeurs physiques telles que la torsion ou la contrainte de rupture par exemple. Il faudrait aussi ajouter du retour d'effort dans le traitement des vaisseaux.

Chapitre 6

Mise en œuvre

Sommaire

La mise en œuvre effective d'un simulateur de chirurgie en informatique est une source de problèmes qui pourrait largement occuper, à elle seule, un ingénieur pendant plusieurs années.

6.1 Le logiciel yav++

6.1.1 Description générale

Le logiciel `yav++`, développé à Epidaure, propose un certain nombre d'algorithmes et de classes de bases formant un environnement de développement à un grand nombre de projets développés au sein d'Epidaure. Ce logiciel est écrit dans le langage `C++` et dispose d'une interface graphique ainsi que d'un interpréteur de scripts basé sur le langage `TCL`. Un grand nombre de systèmes sont supportés par `yav++` : Linux, Windows, Irix, Digital Unix, SunOS. Le simulateur lui-même fonctionne principalement sous Linux mais peut aussi être exécuté sous Irix ou sous Windows. Les différentes fonctionnalités du programme `yav++` sont disponibles sous la forme de modules indépendants qui peuvent être chargés dynamiquement en mémoire en fonction des besoins rencontrés. Les principaux modules liés au simulateur sont :

libModule : Module de bas niveau chargé de l'interfaçage utilisateur et de la gestion du chargement des autres modules ainsi que d'un certain nombre d'autres fonctionnalités (ex : fonctions d'algèbres linéaires de base).

libGraphics : Module en charge des aspects graphiques. Définit la notion de *caméra* (en 2 et en 3 dimensions) et celle de *scène*.

libContour : Module chargé de la gestion des contours (en 2 et en 3 dimensions)

libTriangulation : Module chargé de la gestion de maillages surfaciques triangulaires. Il propose une classe de triangulation statique (`Triangulation3D`) et deux classes de triangulation déformable (`ActiveTriangulation3D` et `PrecomputedTriangulation3D`).

libTetrahedrisation : Module chargé des maillages tétraédriques. Il propose une classe de tétraédrisation statique (`Tetra3D`) et une classe de tétraédrisation déformable (`ActiveTetra3D`).

libForceFeedback : Module chargé de l'interfaçage haut et bas niveaux des systèmes à retour d'effort. Sont concernés pour l'instant le Laparoscopique Impulse Engine de Immersion Corp. et le PhantoM de Sensable. Le module propose de plus une classe simulant la présence de ces périphériques à retour d'effort. Les positions sont alors lues dans un fichier préalablement généré, par exemple par un enregistrement des déplacements de ces périphériques.

libSimulation : Module mettant en œuvre le simulateur de chirurgie.

Un grand nombre d'autres modules existent également (il y en a plus d'une trentaine en tout). On peut citer par exemple, `libInrimage` qui gère des images volumiques et fournit des algorithmes sur ces images (calcul de gradient, seuillage, coupe, etc.),

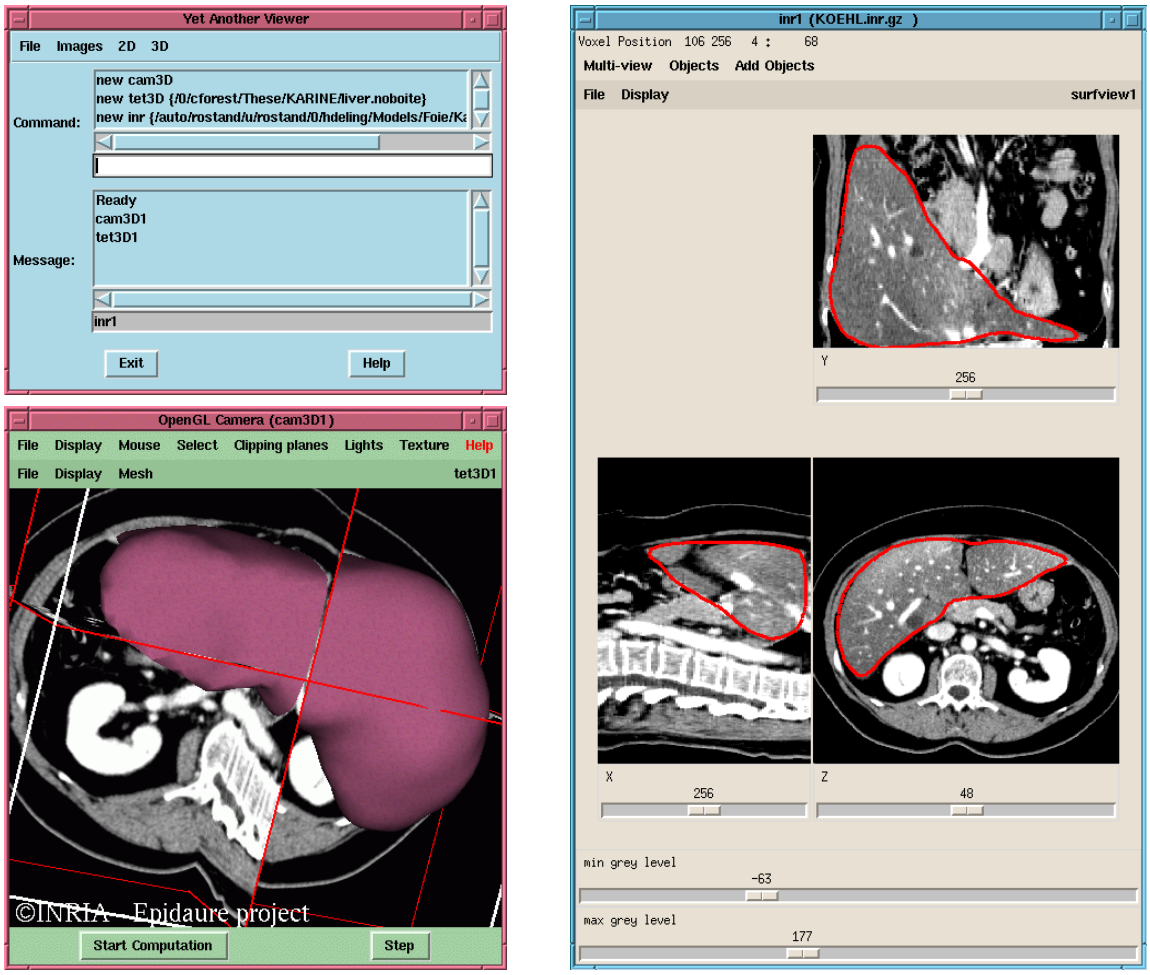


FIG. 6.1 – *Exemple de fenêtres de l'application* **yav++**. *Une image 3D peut être visualisée, au choix, dans une fenêtre dédiée, ou plongée dans une scène 3D. Réciproquement, la trace des maillages de la scène peut apparaître sur l'image. En haut à gauche, l'interpréteur de commandes en ligne.*

`libSM2` qui s'occupe des maillages simplexes, `libCIsosurf` qui calcule une isosurface sur une image, etc.

Regrouper l'ensemble des travaux d'un projet tel que EPIDAURE à l'intérieur d'un seul logiciel permet à chacun d'économiser un temps précieux. Les méthodes permettant la visualisation d'une scène ou l'interfaçage avec le langage `TCL`, les classes simples telles que les vecteurs, les minuteurs, les pointeurs intelligents, les segments de mémoire partagée, etc., ont déjà été écrites une fois pour toutes. De plus, on peut profiter des contributions des autres membres du projet afin d'élargir le champ d'application de ses propres travaux [Sermesant *et al.*, 2002] (cf. figure 6.1).

La contrepartie est que les contributions de chacun doivent pouvoir être utilisées par tous. La maintenance simultanée sous plusieurs plateformes d'un logiciel tel que `yav++` ne peut cependant pas se faire sans certaines contraintes de programmation.

Le langage `C++` n'est pas encore totalement standardisé. Le programme doit donc être écrit à l'intérieur de la partie commune aux "dialectes" des différents compilateurs. Il n'est cependant pas toujours possible de se restreindre à cette intersection. La différence est particulièrement sensible avec le système Windows, le langage `Visual-C++` diffère du `C++-ANSI` dans de nombreux cas, en particulier pour tout ce qui touche la programmation générique. De plus, le chargement dynamique de modules s'effectue d'une manière totalement différente de celle utilisée dans les systèmes de type Unix. À ces différences s'ajoutent des spécificités dues au système d'exploitation ou à la machine elle-même. La gestion des périphériques est, par exemple, effectuée de manière particulière à chaque plateforme. Sur la plupart des machines, le codage interne des flottants utilise la méthode *little-endian*, alors que pour les processeurs MIPS utilisés par les machines SGI c'est la méthode *big-endian* qui est utilisée. Les appels systèmes `IPC` utilisés dans la gestion des segments de mémoire partagés n'existent que sur certains systèmes etc. On a donc eu recours aux facilités proposées par le préprocesseur du langage `C++` pour fournir, lorsque cela est nécessaire, plusieurs versions pour le même morceau de code. Le préprocesseur est aussi utilisé pour masquer le segment de code ayant trait au graphique ou à l'interface `TCL` lorsque l'on désire une version de `yav+++` pouvant fonctionner comme simple bibliothèque de fonctions.

6.1.2 Organisation de yav++

Un certain nombre des objets de `yav++` sont dits *actifs*. Ces objets possèdent une fonction nommée `activation` qui est appelée régulièrement par la boucle principale du programme. L'objet actif le plus important est la *scène*. Une scène regroupe un certain nombre d'objets (contours, maillages simplexes, triangulations, tétraédrisations, etc.) pouvant être visualisés à l'aide d'une caméra. On parlera de *scène 2D* ou de *scène 3D* selon que les objets sont plongés dans $\mathbb{R}^2$ ou dans $\mathbb{R}^3$. Pour visualiser une scène, on utilise un objet appelé *caméra* ; il peut y avoir plusieurs caméras pour visualiser une même scène. Certains des objets de la scène sont dits *déformables*. Ces objets disposent d'une fonction `iterate`, chargée du calcul des déformations et appelée à chaque exécution de la fonction `activation` de la scène. Le détail du calcul des déformations n'est pas imposé mais il existe un mécanisme qui est employé à la fois par les maillages déformables triangulaires, tétraédriques et simplexes ainsi que par les contours déformables.

Chaque objet dispose d'un pointeur sur une *force interne*. Il s'agit en fait d'une structure regroupant les méthodes chargées du calcul des forces internes pour un modèle donné (masse-ressort, masse-tenseur linéaire ou non linéaire, etc.). Cette structure peut éventuellement fournir des fonctions permettant le calcul des forces sommet par sommet, ainsi que celles chargées de l'initialisation globale ou locale de ces forces. Le calcul sommet par sommet est utile dans le cas d'une résolution asynchrone (cf. plus loin § 6.2) ou dans celui d'un maillage mixte (cf. § 2.9). L'initialisation locale est utile lors des opérations de découpe (cf. § 2.7.2).

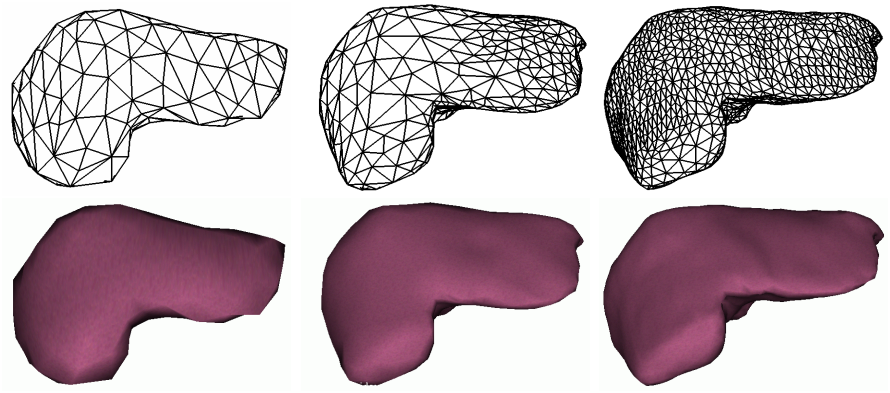


FIG. 6.2 – *Trois résolutions différentes du même foie : 350 tétraèdres, 900 tétraèdres et 3300 tétraèdres.*

6.2 Calculs de déformation Temps-Réel : résolution asynchrone

C'est sur le calcul des déformations que le programme passe le plus de temps. Pour mesurer l'efficacité et le réalisme d'une méthode de calcul, on dispose de deux grandeurs. La première est la rapidité, qui indique en combien de temps la méthode se termine. La seconde est la précision qui caractérise le réalisme de la solution proposée. Pour être admissible dans le cadre du simulateur, une méthode donnée doit être suffisamment rapide pour permettre au moins une vingtaine de rafraîchissements de l'image par seconde pour des maillages d'environ 2000 à 3000 tétraèdres. Cela permet de garantir la fluidité de l'animation. Pour augmenter la rapidité d'une méthode, la solution que nous avons employée consiste simplement à restreindre le nombre d'éléments du maillage. Pour être visuellement réaliste, il n'est en effet nul besoin d'utiliser des maillages de centaines de milliers d'éléments. Pour un foie, un maillage de 300 tétraèdres bien texturé est à peu près aussi réaliste qu'un maillage de 4.000 tétraèdres. La méthode de raffinement dynamique proposée au chapitre 3 permet de changer *à la volée* la précision du maillage aux endroits où cela se révèle être nécessaire. La précision de la résolution s'estime par comparaison à une solution de référence. C'est d'ailleurs le sens de la collaboration de la société ESI lors du projet CAESARE que de fournir ce modèle de référence. On peut aussi faire appel à des estimations quantitatives fournies par l'expérience (trop dur, trop mou, trop élastique, etc.). Les discussions sur le meilleur modèle de déformation à utiliser en simulation de chirurgie sont encore actives et font l'objet de publications régulières [Costa and Balaniuk, 2001; Schwartz *et al.*, 2002; Balaniuk and Salisbury, 2003]. Nous proposons ici une réflexion sur la manière d'accélérer la convergence de la plupart de ces méthodes.

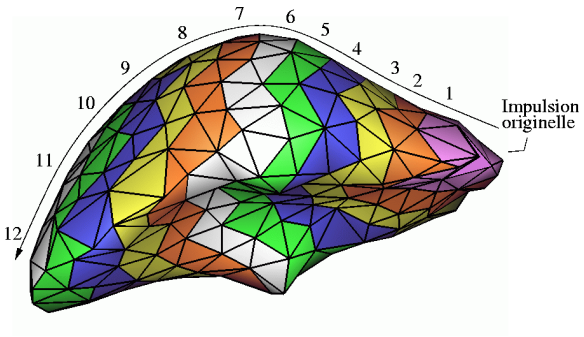


FIG. 6.3 – *Distance par rapport au tétraèdre rouge (à droite). Une impulsion sur ce tétraèdre mettrait 12 itérations pour se propager dans la totalité du maillage.*

6.2.1 Généralités, résolution synchrone

Le calcul direct du champ de déplacement $U_{\mathcal{S},\mathcal{F}}$ des sommets $\mathcal{S}$ du maillage résultant de l'application d'un certain nombre de forces $\mathcal{F}$ se révèle souvent assez long, car il peut nécessiter la résolution d'un système matriciel de grande taille. Un grand nombre de méthodes de résolution préfèrent donc procéder de manière itérative. Cela consiste à s'approcher de la solution en effectuant plusieurs fois des calculs du type :

$$U_{\mathcal{S},n+1} = F(U_{\mathcal{S},n},\mathcal{F})$$

ou encore :

$$U_{\mathcal{S},n+1} = F(U_{\mathcal{S},n},U_{\mathcal{S},n-1},\mathcal{F})$$

où $U_{\mathcal{S},n}$ représente le champ de déplacement des sommets $\mathcal{S}$ obtenu à l'itération n, et F est une fonction dépendant du modèle mécanique utilisé et de la topologie du maillage et qui vérifie :

$$U_{\mathcal{S},n} \underset{n\to\infty}{\longrightarrow} U_{\mathcal{S},\mathcal{F}}$$

Ces méthodes sont intéressantes car, chaque itération étant beaucoup moins coûteuse que le calcul direct, elles permettent d'obtenir très rapidement une approximation de la solution. À chaque itération, ces méthodes calculent un nouveau déplacement *pour l'ensemble des sommets du maillage*. Les positions des sommets étant mis à jour en même temps on dit qu'elles procèdent de manière *synchrone*. Évidement, plus le nombre d'itérations est élevé, plus la solution obtenue est précise. Pour le simulateur, et compte tenu des contraintes de rapidité, seules une ou deux itérations peuvent être effectuées à chaque pas de temps, ce qui peut donner aux déformations un aspect visqueux peu réaliste. En effet, les déformations se propagent lentement à l'intérieur du maillage, à savoir un tétraèdre par itération (voir figure 6.3).

Les méthodes que nous proposons consistent à résoudre le champ de déplacement, non plus dans sa globalité, mais sommet par sommet. Le champ $U_{\mathcal{S},n+1}$ ne différera du champ $U_{\mathcal{S},n}$ que par la position d'un seul sommet. On dit qu'on procède alors de manière *asynchrone*. Cette méthode est semblable à une résolution par la méthode de Gauss-Seidel [Désidéri, 1998] dans le cadre de la théorie de décomposition de

domaines. La convergence est acquise lorsqu'il s'agit d'un problème de minimisation et que chaque itération diminue la valeur globale de la grandeur à minimiser. Dans notre cas, il s'agit généralement de l'énergie élastique. La vitesse de convergence et son intérêt par rapport à la résolution synchrone dépendra cependant du type de problème et de la manière avec laquelle il est résolu. Potentiellement, cette méthode présente plusieurs intérêts :

- concentrer les calculs sur les zones présentant des déformations pertinentes et éviter de perdre du temps sur des sommets ne se déplaçant que très peu ou déjà à l'équilibre,
- contrôler plus finement le temps passé à calculer les déformations et être capable d'arrêter les calculs par exemple au bout d'un trentième de seconde,
- augmenter la vitesse de propagation des déformations en jouant sur l'ordre dans lequel sont traités les sommets.

6.2.2 Itérations préférentielles

Concept

L'idée de cette méthode est de résoudre un par un les sommets de manière aléatoire, en se débrouillant pour que ceux des zones fortement déformées soient choisis plus fréquemment. À chaque sommet est affectée une grandeur entière appelée *priorité*, proportionnelle à la probabilité qu'a ce sommet d'être tiré aléatoirement. On notera p_A la priorité du sommet A et $P(A)$ la probabilité qu'il a d'être choisi.

Gestion des tirages aléatoires

Le premier problème à résoudre est de se donner les moyens d'effectuer, parmi les sommets, à un tirage aléatoire pondéré par leur priorité.

$$\forall A,B \ \in \mathcal{S},\ \frac{P(A)}{P(B)} = \frac{p_A}{p_B}$$

Pour cela, on crée un vecteur suffisamment long appelé *queue de priorité*. Chaque élément de ce vecteur est un pointeur vers l'un des sommets du maillage et il y a autant de pointeurs sur un sommet donné que la valeur de la priorité de ce sommet. On opère alors un tirage aléatoire parmi les éléments de ce vecteur. De plus, chaque sommet mémorise les indices des éléments du vecteur qui correspondent à son pointeur. Lorsque la priorité d'un sommet augmente, il rajoute à la fin du vecteur autant de fois son pointeur que cela est nécessaire. La taille du vecteur est augmentée en conséquence. Lorsque sa priorité diminue, il va effacer du vecteur le nombre nécessaire d'itérations de son pointeur et recopier à la place les éléments situés à la fin du vecteur. La taille du vecteur est diminuée en conséquence.

Toutes les priorités valent 3

0	0	0	1	1	1	2	2	2	3	3	3

La priorité du sommet 0 passe de 3 à 4

0	0	0	1	1	1	2	2	2	3	3	3	**0**

La priorité du sommet 1 passe de 3 à 1

0	0	0	1	**3**	**0**	2	2	2	3	3

TAB. 6.1 – *Gestion de la queue de priorité*

TAB. 6.2 – *Valeurs de $\overline{\mathcal{P}}(A)$ en fonction de p_A/L.*

p_A/L	0,5	1	2	3	5	10	20	30
$\overline{\mathcal{P}}(A)$	0,61	0,37	0,14	0,050	$6{,}7\ 10^{-3}$	$4{,}5\ 10^{-5}$	$2{,}1\ 10^{-9}$	$9{,}4\ 10^{-14}$

Étude statistique

On s'intéresse à la probabilité pour un sommet de ne pas être itéré au cours d'un pas de simulation donné. Cette probabilité est importante car si l'un des sommets d'une zone en train de se déformer n'est pas itéré durant un pas de temps donné, il donnera l'impression que le maillage se déforme de manière haché.

Dans l'état actuel des puissances de calcul et pour les maillages utilisés, un pas de simulation consiste en n résolutions de sommets, n étant le nombre de sommets du maillage. Soit l la longueur de la queue de priorité. Posons $L = l/n$. Si p_A est la priorité du sommet A, la priorité pour ce sommet d'être choisi au cours d'un tirage donné vaut :

$$P(A) = p_A/l$$

En supposant qu'aucune priorité ne change au cours du pas de temps, la probabilité pour que ce sommet ne soit jamais choisi vaut :

$$\overline{\mathcal{P}}(A) = (1 - p_A/(L.n))^n \underset{n\to\infty}{\longrightarrow} e^{-p_a/L}$$

Ce résultat à l'asymptote est intéressant car il montre que seul le rapport p_a/L importe. Il n'est donc pas nécessaire d'avoir des queues de priorité trop longues. En fait, si on impose que chaque sommet ait une priorité minimum de 1 - de façon à ce que l'on oublie pas systématiquement un sommet - on peut se contenter d'une queue de priorité de longueur égale à $2.n$.

On peut différencier les sommets en deux catégories : ceux dont on est à peu près sûr qu'ils seront itérés au cours du pas de simulation et ceux dont l'itération n'est pas assurée. Si on suppose que les parties visibles sont composées d'une centaine de sommets, que la fréquence d'itération est d'une trentaine de hertz, et si on désire ne pas avoir plus d'un problème (ie. sommet actif visible non itéré durant un pas d'itération) par dizaine de secondes, on peut situer la limite entre les deux catégories

de sommets à une valeur de $\overline{\mathcal{P}}(A)$ d'environ 10^{-5}. Un regard sur les valeurs prises par $\overline{\mathcal{P}}(A)$ (cf. table 6.2) nous indique que cela correspond à une valeur de p_A/L proche de 10. D'autre part, lorsque L vaut 2, un sommet de priorité égale à 1 ne sera itéré, en moyenne, qu'une fois sur trois.

Estimation de la priorité

La priorité caractérise la pertinence d'un sommet donné pour les calculs de déformation. On suppose que cette pertinence est directement liée à la valeur ΔA du déplacement de ce sommet au cours de la dernière itération le concernant. Cette supposition est cependant critiquable car si un sommet est itéré plusieurs fois de suite trop rapidement, celui-ci atteint une position d'équilibre local et ne se déplace donc plus. On verra plus loin comment on s'affranchit de ce cas. On calcule la priorité à l'aide d'une formule du type :

$$P(A) = E(F(\Delta A,C))$$

où E représente l'opérateur *partie entière* et où C est une constante correspondant à l'*excitation* du maillage et qui est régulièrement ajustée en fonction de la taille atteinte par la queue de priorité. Si la taille du vecteur devient trop grande, on augmente la valeur de C. Si cette taille devient trop petite, on diminue cette valeur. F est une fonction discriminante qui permet de différencier les sommets pertinents des sommets non pertinents. Typiquement, on prend une fonction de la forme :

$$F = \alpha . \left(\frac{\Delta A}{C}\right)^{\beta}$$

Un sommet de priorité faible situé à proximité d'un sommet de priorité élevé est lui aussi susceptible de se déplacer. Lorsque la priorité d'un sommet augmente de manière significative, on va donc augmenter sensiblement la priorité de ses sommets voisins. Sans ce dispositif, un tel sommet de priorité originelle très faible pourrait ne pas être itéré avant plusieurs pas de simulation. Inversement, on va empêcher la priorité de diminuer trop fortement du fait d'une réestimation, de façon à se protéger des cas où un sommet de priorité très forte serait itéré plusieurs fois de suite.

Résultats

La méthode améliore très sensiblement la réactivité du maillage (voir figures 6.4 et 6.5) et, en ce sens, elle semble très prometteuse. On l'a cependant abandonnée car on n'est jamais arrivé à s'affranchir complètement d'une certaine irrégularité dans les déformations du maillage due au fait que certains sommets ne s'itèrent pas suffisamment par rapport à leurs voisins. En outre, pour un nombre égal de sommets traités, cette méthode est environ 30% plus lente que la méthode synchrone (voir figure 6.6). Cela est dû au surcoût causé par le traitement des priorités. Malgré cela, elle reste beaucoup plus rapide que la méthode synchrone. De plus, la possibilité de pouvoir

contrôler précisément le temps passé dans le calcul des déformations peut être une caractéristique intéressante pour la garantie du temps réel.

6.2.3 Ordonnancement

Concept

Cette méthode s'inspire des travaux de Joel Brown réalisés à Stanford dans le cadre du projet de microchirurgie [Brown *et al.*, 2001b]. L'idée de Brown est de jouer sur l'ordre de résolution des sommets d'un maillage surfacique pour augmenter la vitesse de convergence des calculs. La première étape de son algorithme consiste donc à ordonner les sommets du maillage. Les sommets de *niveau 0* sont ceux qui sont directement soumis à une force extérieure, généralement causée par l'action de l'outil de l'utilisateur. Les sommets de niveau 1 sont les sommets adjacents à ces sommets de niveau 0, les sommets de niveau 2 ceux adjacents aux sommets de niveau 1, et ainsi de suite, en cercles concentriques, jusqu'à recouvrir la totalité du maillage. Les sommets de niveau 0 sont alors résolus en premier, puis, successivement, les sommets de niveau 1, ceux de niveau 2, etc. Brown emploie une technique supplémentaire pour accélérer encore la convergence. Plusieurs itérations sont effectuées au cours d'un même pas de temps et si, au cours d'une itération, les positions de la totalité des sommets d'un niveau k donné sont modifiées moins qu'une valeur seuil définie à l'avance, alors l'itération est interrompue et une autre itération est relancée. Cette méthode apparaît très efficace lorsque les déformations ont un caractère local mais n'est applicable que si on garantit que l'équilibre mécanique est obtenue à la fin de chaque pas de temps.

Nous avons en fait essayé plusieurs méthodes pour déterminer l'ordre dans lequel sont résolus les sommets. La première est une application directe de la méthode proposée par Brown au cas d'un maillage volumique. Si aucune force ne s'exerce sur les sommets du maillage, on garde l'ordre de résolution utilisé pour le pas de temps précédent. On ne s'est pas permis d'interrompre certaines itérations car l'expérience montre que l'équilibre n'est jamais atteint à la fin d'un pas de temps. La deuxième méthode consiste simplement à traiter les sommets dans un ordre aléatoire. L'intérêt de cette dernière méthode est de ne nécessiter ni la détermination des sommets sur lesquels s'exercent une force, ni les calculs de propagation de la résolution.

Résultats, rapidité de convergence

Les figures 6.4, 6.5 et 6.6 illustrent les premiers résultats des comparaisons entre ces méthodes et la méthode synchrone. Dans ces expériences, on déforme d'abord le maillage à l'aide de l'outil virtuel, puis on observe la vitesse avec laquelle il retourne dans son état au repos. Une première remarque est que ces méthodes sont un peu plus rapides que la méthode synchrone (cf. figure 6.6). Cette différence marginale est due à des différences dans le parcours des objets du maillage. La méthode synchrone parcourt en effet l'ensemble des arêtes et des sommets du maillage pour calculer les

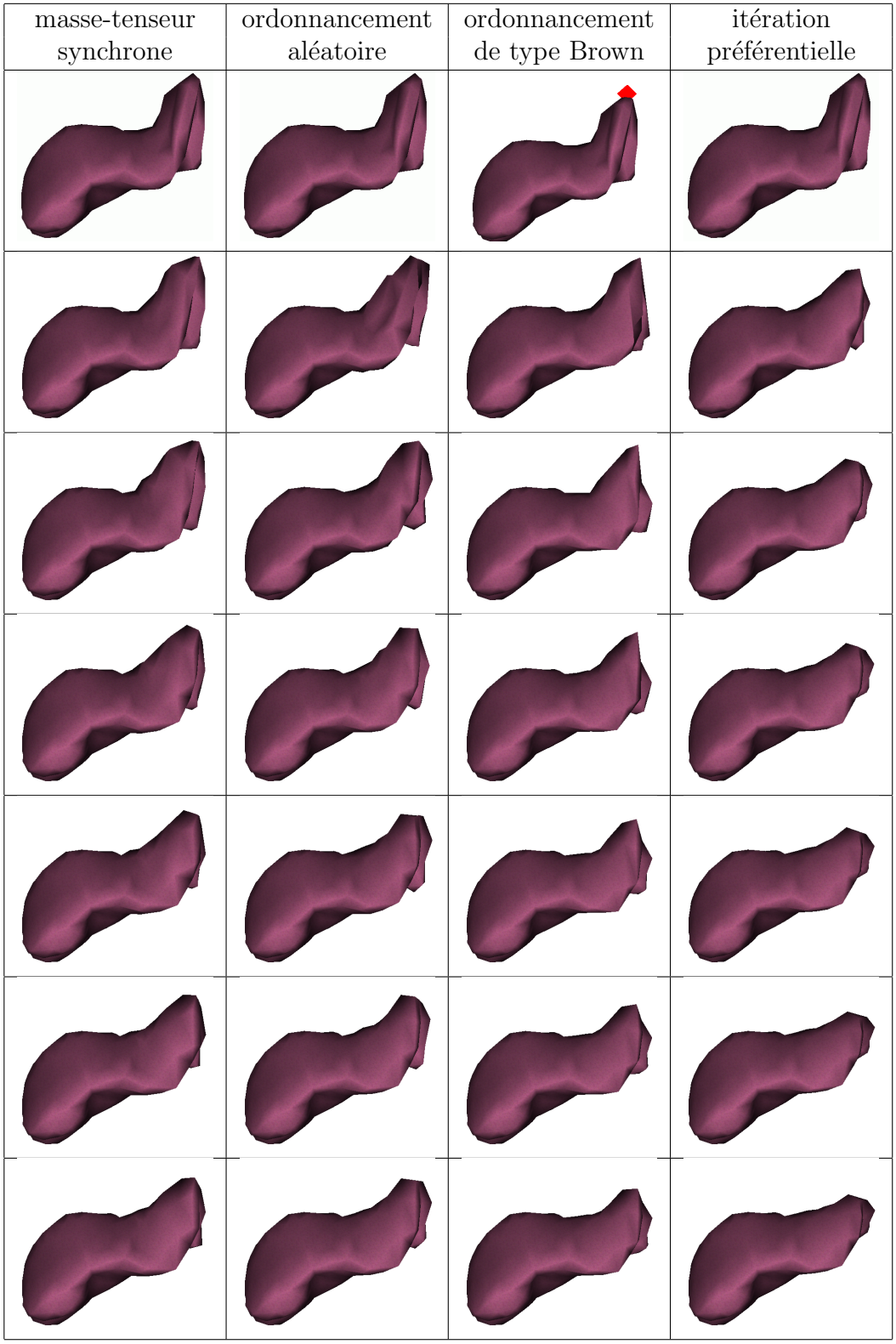


FIG. 6.4 – *Comparaison de l'évolution des déformations pour les différentes méthodes de résolution asynchrone au cours des 6 premiers pas de temps. Pour l'ordonnancement de type Brown, le sommet indiqué en rouge sur la première image est marqué comme ayant subi une force externe.*

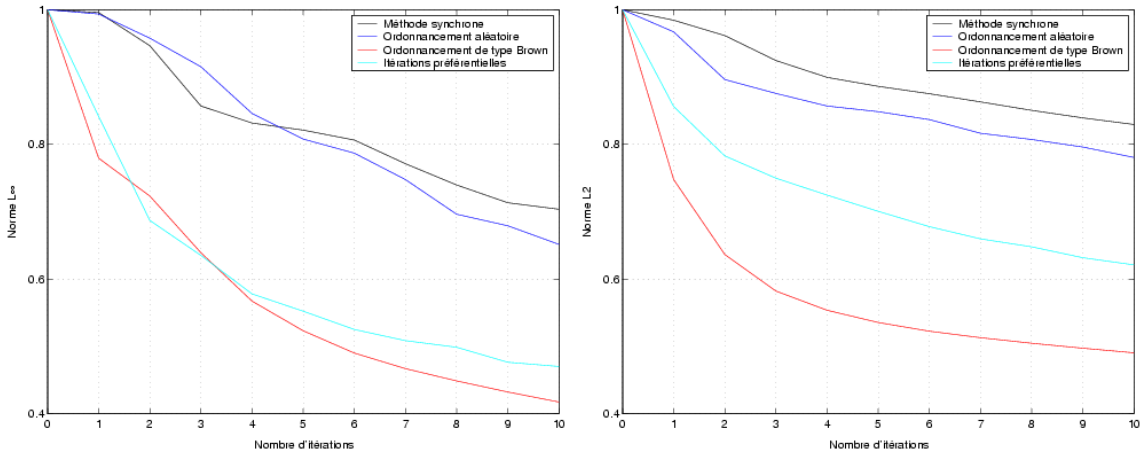


FIG. 6.5 – *Comparaison de la convergence des différentes méthodes proposées pour les normes L_∞ et L_2 lors de l'expérience représentée figure 6.4*

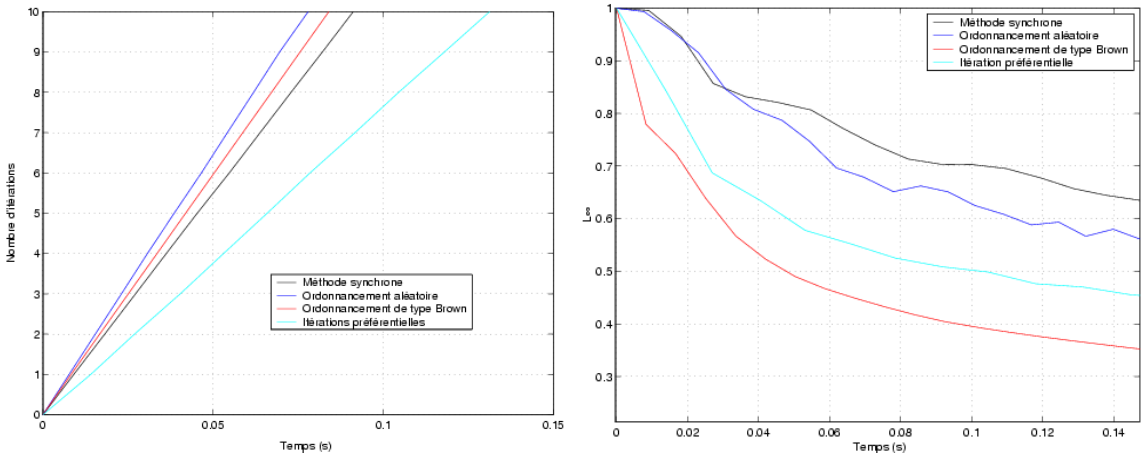


FIG. 6.6 – *Comparaison des vitesses d'exécution des méthodes proposées. Comparaison de la convergence L_∞ en fonction du temps*

forces, puis de nouveau l'ensemble des sommets afin de procéder aux déplacements, la méthode asynchrone quant à elle parcourt une seule fois les sommets et, pour chacun d'eux, les arêtes adjacentes.

Si la méthode d'ordonnancement aléatoire se révèle décevante car à peine meilleure que le cas synchrone, ce n'est pas le cas pour la méthode inspirée des travaux de Brown. La vitesse de convergence de celle-ci est en effet beaucoup plus grande que pour le cas synchrone et même plus grande que la méthode d'itérations préférentielles présentée précédemment. Cette méthode n'est cependant pas utilisée dans le simulateur car elle entraîne encore certaines irrégularités dans la déformation du maillage. Elle est cependant très prometteuse car efficace et relativement simple à mettre en œuvre.

6.3 Gestion des périphériques à retour d'effort

6.3.1 Gestionnaires de périphérique

Les gestionnaires de périphérique sont des programmes qui réalisent l'interface entre le programmeur et un périphérique donné. Il existe un nombre insoupçonné de tels programmes. On connaît bien le gestionnaire de carte graphique, celui du lecteur de CD-ROM, celui de l'imprimante ou du modem, mais il en existe aussi pour le clavier, pour la souris, pour les haut-parleurs, etc. Sous les systèmes de type Unix, une visualisation du répertoire `/dev` (périphérique se dit *device* en anglais), qui regroupe l'ensemble des gestionnaires, montre qu'un tel système en possède un nombre très grand (entre 200 et 700 suivant les machines du laboratoire). Ce type de programme est aussi utilisé pour interfacer les accès à la mémoire vive de la machine et aux différents disques, mais aussi aux différentes consoles virtuelles, aux ports séries et USB, aux bus PCI et SCSI, etc. Il existe plusieurs raisons principales qui poussent à passer à travers un programme tiers pour accéder aux périphériques.

Facilité d'utilisation Ces programmes fournissent un ensemble de méthodes traduisant, dans le langage interne du périphérique, les instructions envoyées par l'utilisateur. La tâche de celui-ci est donc facilitée.

Protection du matériel L'utilisation systématique de ces méthodes a aussi l'intérêt de protéger le périphérique d'un certain nombre d'utilisations maladroites ou non conformes au protocole.

Sécurité Ces programmes sont aussi utiles dans la gestion des conflits et des droits d'accès. En effet, une imprimante par exemple ne peut pas être utilisée simultanément par plusieurs utilisateurs et il n'est généralement pas souhaitable qu'elle puisse l'être par des utilisateurs extérieurs ou non autorisés.

On peut différencier trois niveaux principaux dans le fonctionnement d'un ordinateur. *Le niveau matériel*, composé du processeur, de la mémoire, des disques, des ports externes, etc. *Le niveau système*, composé d'un ensemble de programmes chargés de la gestion du niveau matériel et que l'on appelle généralement *noyau*. *Le*

niveau utilisateur, qui contient tous les autres programmes. Le parti pris par les systèmes d'exploitation modernes [Beauquier et Bérard, 1994] est de limiter l'accès au niveau matériel aux seuls programmes du niveau système. Les vieux systèmes mono utilisateur (MS/DOS, CP/M, etc.) ne pratiquaient pas cette distinction : l'utilisateur pouvait sans aucun problème accéder au matériel et, par exemple déplacer à volonté la tête de lecture du disque dur ou contrôler le balayage du faisceau de l'écran. Les programmes du niveau système ne sont accessibles à partir du niveau de l'utilisateur qu'à travers un certain nombre de passerelles appelées *appels système*. Le *super-utilisateur* est un utilisateur particulier qui a la possibilité d'accéder sans restriction à l'ensemble des programmes du niveau système.

Comme un grand nombre d'autres gestionnaires de périphériques, le gestionnaire fourni par la société Immersion pour l'utilisation du *Laparoscopic Impules Engine*(LIE) se compose de deux parties, l'une située au niveau système, l'autre au niveau utilisateur. La partie système permet d'atteindre les registres de la carte et de les rendre accessibles à l'utilisateur. La partie utilisateur fournit un certain nombre de méthodes facilitant l'usage du périphérique : `Init()`, `ReadEncChan`, `DACOut`, etc. On peut remarquer d'une part qu'il n'existe aucun mécanisme permettant d'empêcher les accès simultanés au périphérique, et d'autre part que les fonctions fournies à l'utilisateur sont encore relativement bas niveau. On devra donc prévoir une couche supplémentaire pour pouvoir réellement utiliser le périphérique.

Le gestionnaire d'Immersion est disponible pour les systèmes d'exploitation MS-DOS, Windows xx, et IRIX. Le simulateur tournant principalement sous Linux (même s'il peut aussi fonctionner sous IRIX et sous Windows), il a été nécessaire de concevoir le gestionnaire de périphérique correspondant.

Le gestionnaire proposé par l'entreprise Sensable pour la gestion du PhantoM est beaucoup plus fourni. En particulier, le niveau utilisateur se compose d'une bibliothèque complète appelée GHOST qui permet d'utiliser rapidement et simplement le périphérique. En quelques dizaines de lignes, il est ainsi possible de lire les positions et de créer un champ de force de forme arbitraire. Cette bibliothèque fonctionne sous Windows et depuis peu sous Linux. La version sous Irix n'est plus maintenue.

6.3.2 Gestionnaire de périphérique pour Linux

Le système d'exploitation Linux, à l'image des systèmes de type UNIX, considère l'ensemble des opérations d'accès aux périphériques de la même manière que s'il s'agissait d'opérations sur des fichiers [Rubini, 1998]. Les périphériques disponibles sont donc matérialisés dans le répertoire `/dev` sous forme de *pseudo-fichiers* qu'il est possible d'ouvrir, de fermer, de lire, etc. La partie bas niveau d'un gestionnaire de périphérique sous Linux consiste donc en la réécriture de ces opérations typiques. On distingue en fait deux familles de périphériques suivant que les données sont transmises caractère par caractère, comme c'est le cas pour un haut parleur ou une console, ou bloc par bloc, comme c'est le cas pour la mémoire vive ou le disque dur. Les gestionnaires de périphériques par bloc sont plus complexes à rédiger que ceux par caractère

car ils doivent, entre autres, prendre en compte la notion d'interruption.

À chaque pseudo-fichier est affecté un *nombre principal* et un *nombre secondaire*. Le premier indique au système à quel type de périphérique il correspond et quelle surcharge des fonctions classiques d'accès aux fichiers il devra utiliser. Le pseudo-fichier `/proc/devices` liste les nombres principaux déjà affectés. Le nombre secondaire est transmis au gestionnaire comme argument lors de l'ouverture et peut servir à distinguer deux périphériques différents ou deux manières d'utiliser le même périphérique (voir table 6.3).

Nous avons décidé d'utiliser le nombre secondaire afin de différencier les différents LIE branchés sur une machine. La valeur 0 correspond au premier LIE, la valeur de 1 au second, etc.

Les LIE sont branchés sur des ports PCI. Il existait anciennement des modèles branchés sur des cartes ISA mais ceux-ci ont quasiment disparus. Les périphériques PCI sont identifiables grâce à deux valeurs appelées VendorID et DeviceID. La première identifie le fabriquant et la deuxième le produit, ce qui permet de retrouver facilement un périphérique connu parmi tous ceux disponibles. À noter cependant que la carte PCI utilisée par le PhantoM est identique à celle utilisée par le LIE. Si jamais les deux systèmes sont branchés sur la même machine, il faudra faire appel à d'autres techniques pour les différencier.

Les fonctions surchargeables sont nombreuses et, pour un périphérique donné, toutes n'ont pas besoin d'être présentes. Dans notre cas, on se contente de surcharger les cinq fonctions suivantes :

`open` Cette fonction, appelée par l'instruction `open`, avertit le gestionnaire de l'ouverture et de la fermeture d'un pseudo-fichier correspondant à un LIE. Lorsqu'il y a plusieurs LIE sur une même machine, on se sert du nombre secondaire pour déterminer lequel. Le système associe une zone de la mémoire virtuelle à chacune des zones adressables de la carte. Les spécifications du LIE fournissant la position des registres par rapport au début de ces zones adressables, on doit donc mémoriser leur localisation.

`release` Cette fonction est appelée lorsqu'on ferme le pseudo-fichier à travers l'instruction `close`. Elle libère les ressources utilisées.

`lseek` Détermine l'adresse où s'effectuera la prochaine opération de lecture/écriture. Est appelée à travers l'instruction `seek`.

`read/ write` Opérations de lecture et d'écriture.

Lorsqu'elle n'est pas directement codée dans le noyau, l'introduction d'un nouveau périphérique se fait à l'aide de *modules dynamiques*. Les modules sont des bouts de code qui permettent de rajouter un certain nombre de nouvelles fonctionnalités au système. Le module est chargé en mémoire au démarrage ou par un super-utilisateur à l'aide de la commande `insmod`. Il rajoute alors au noyau les symboles correspondant aux fonctions surchargées. Ces fonctions sont alors affectées à un nombre principal à l'aide de la fonction `register_chrdev`. Au besoin, ce nombre peut être désaffecté avec la commande `unregister_chrdev`.

Principal	Secondaire	Nom	Description
1	1	/dev/mem	Accès à la mémoire physique
	2	/dev/kmem	Accès à la mémoire virtuelle du noyau
	3	/dev/null	Périphérique nul
2	0	/dev/fd0	Premier lecteur de disquette
	1	/dev/fd1	Deuxième lecteur de disquette
3	0	/dev/hda	Disque maître
	1	/dev/hda1	Première partition du disque maître
	64	/dev/hdb	Disque esclave
4	0	/dev/console	Console
	1	/dev/tty1	Première console virtuelle
	64	/dev/ttyS1	Premier port série
	128	/dev/ptyp0	Premier tty virtuel maître
	192	/dev/ttyp0	Premier tty virtuel esclave
6	0	/dev/lp0	Première imprimante parallèle
	1	/dev/lp1	Deuxième imprimante parallèle
7	0	/dev/vcs	Accès texte à la console virtuelle courante
	1	/dev/vcs1	Accès texte à la première console virtuelle
8	0	/dev/sda	Premier disque SCSI
	1	/dev/sda	Premier partition du premier disque SCSI
	16	/dev/sda	Deuxième disque SCSI
10	0	/dev/logibm	Souris Logitech
	1	/dev/psaux	Souris PS/2
11	0	/dev/scd0	Premier lecteur de CD-ROM SCSI
14	0	/dev/mixer	Réglage de volume
	2	/dev/midi00	Premier port MIDI
	3	/dev/dsp	Audio digital
15	0	/dev/js0	Premier Joystick
	0	/dev/js1	Deuxième Joystick

TAB. 6.3 – *Conventions Linux sur la répartition des premiers nombres principaux et secondaires pour les gestionnaires de périphériques.*

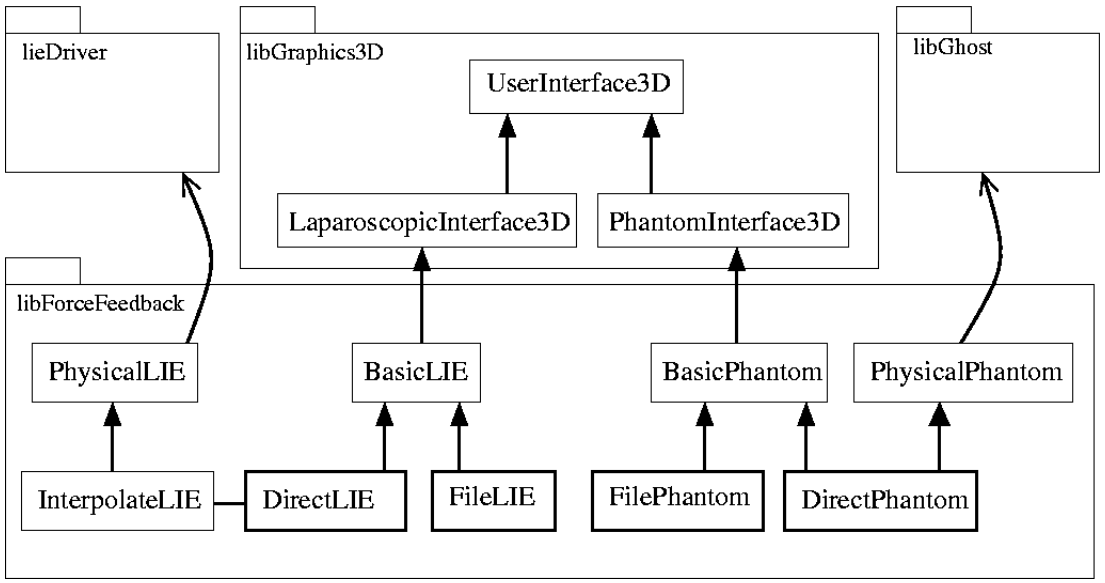


FIG. 6.7 – *Hiérarchies des interfaces des systèmes à retour d'effort de* `yav++`*. Les objets directement utilisables par l'utilisateur sont entourés en gras.*

6.3.3 Interface des systèmes à retour d'effort

Nous avons développé une interface semblable permettant d'utiliser de manière transparente les deux systèmes à retour d'effort dont nous disposons : le Laparoscopic Impulse Engine et le PhantoM. Ces interfaces sont regroupées dans la bibliothèque `libForceFeedback`. L'organisation de la bibliothèque est schématisée figure 6.7.

On a défini une classe appelée `UserInterface3D` dont sont censés hériter tous les objets pouvant servir d'interface utilisateur. Ces objets sont listés dans chaque scène et peuvent donc être retrouvés facilement au besoin. La bibliothèque `libGraphics3D` propose deux interfaces dérivant de cette classe, une pour chacun des deux types de système à retour d'effort : `LaparoscipicInterface3D` pour les systèmes semblables au LIE (3 degrés de liberté), et `PhantomInterface3D` pour les systèmes semblables au PhantoM (6 degrés de liberté). Ces interfaces contiennent des entrées pour chacune des fonctions que l'on veut voir implémentées : détermination de la position, de l'orientation, envoi de force, etc.

Ces classes ont été placées dans la bibliothèque `libGraphics3D` afin de ne pas rendre dépendantes de `libForceFeedback` des bibliothèques utilisant les systèmes à retour d'effort de manière accessoire. Ces dernières ne connaissent donc pas le détail de la structure de `libForceFeedback` et ne travaillent en fait qu'avec des objets de type `LaparoscipicInterface3D` et `PhantomInterface3D` sans savoir s'il s'agit d'un périphérique réel ou d'un périphérique simulé.

Il existe deux objets principaux dérivant de `PhantomInterface3D`. Le premier se nomme `FilePhantom` et simule la présence d'un PhantoM en allant lire dans un fichier la position et l'orientation que prend l'outil. L'autre se nomme

`DirectPhantom` et sert réellement d'interface au PhantoM. Les positions sont lues grâce à l'objet `PhysicalPhantom` à partir des méthodes de `libGhost`. L'objet `BasicPhantom` regroupe les méthodes d'affichage qui sont communes à `FilePhantom` et à `DirectPhantom`.

La structure de l'interface du LIE est très semblable à celle du PhantoM. Il existe deux objets principaux : `FileLIE` qui lit les positions dans un fichier, et `DirectLIE` qui interface un LIE réel. L'objet `BasicLIE` regroupe les méthodes d'affichage communes à ces deux objets. La différence réside dans le fait que la position du LIE n'est pas lue directement par l'objet `DirectLIE` mais par un autre appelé `InterpolateLIE`. Ce dernier objet est aussi chargé du calcul des forces et est créé dans un processus à part tournant à 500 Hz (voir partie 4.5). Ces deux objets dialoguent à travers des segments de mémoire partagée. Le PhantoM n'a pas besoin d'un tel dispositif car la librairie GHOST intègre déjà des fonctionnalités permettant une génération asynchrone des consignes de force.

6.4 Graphique

La crédibilité du simulateur passe en grande partie par le réalisme des vues générées. Les opérations d'affichage consomment une grande partie du temps de calcul de la machine. Elles sont en concurrence directe avec le calcul des déformations et il faut prendre garde à ne pas trop léser ces derniers. On a donc été conduit à développer plusieurs techniques permettant d'améliorer l'aspect du rendu ou d'optimiser sa vitesse de traitement.

6.4.1 Ajout de texture

Nous nous sommes inspirés de certains travaux de l'équipe iMagis réalisés au cours de l'action AISIM [Neyret *et al.*, 2002]. L'ajout de texture et de texture de lumière permet d'augmenter facilement le réalisme de l'affichage. L'emploi des textures a été rendu possible avec l'introduction dans la structure de données de la notion de zone de surface (voir chapitre 2.6.6). La texture utilisée pour le foie a été obtenue à partir de l'enregistrement vidéo d'une véritable opération. La texture de lumière est une texture classique pour laquelle la valeur des coordonnées de texture dépend de la normale à la surface au point considéré [Székely *et al.*, 2000].

6.4.2 PN-Triangles

Plus grand est le nombre d'éléments de la surface du maillage, plus précise est la géométrie de celui-ci. Malheureusement, plus le maillage est fin, plus ses éléments sont nombreux et plus les calculs de déformation prennent du temps. Les techniques de raffinement dynamique proposées au chapitre 3 permettent de diminuer le nombre

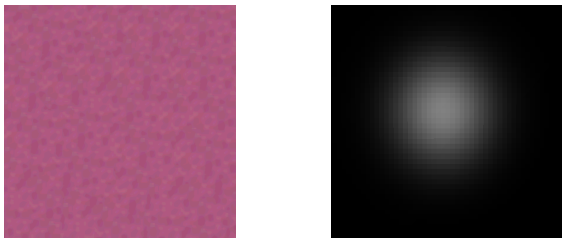


FIG. 6.8 – *Texture utilisée pour le foie. Exemple de texture de lumière.*

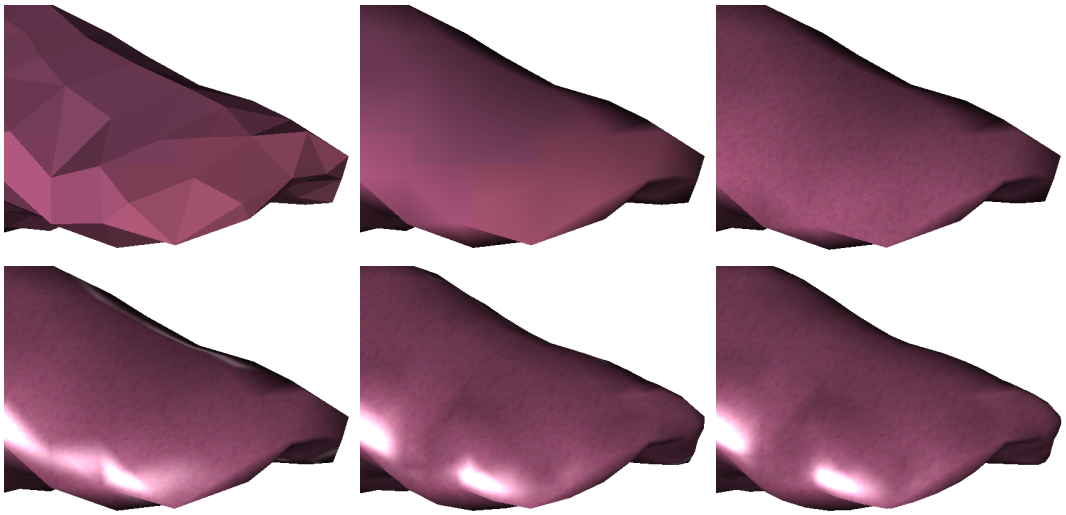


FIG. 6.9 – *Différents modes de rendu. De gauche à droite : plat, Gouraud (lissage des normales), texturé, texture de lumière, PN triangles de niveau de détail de 1 et de 2*

d'éléments du maillage. Cette augmentation de la taille des triangles de la surface risque cependant de diminuer le réalisme de l'affichage. Il existe un certain nombre de techniques permettant de simuler l'utilisation d'un maillage plus fin que ce qu'il est en réalité. L'idée la plus répandue est d'habiller un maillage volumique grossier avec un maillage surfacique plus fin. C'est par exemple ce que fait G.Debunne [Debunne, 2000] pour ses modèles de déformation multirésolution. La position de chacun des sommets du maillage surfacique est déterminée par rapport à la position d'un certain nombre de sommets du modèle volumique. Cette technique permet de déformer très efficacement des modèles visuellement complexes mais mécaniquement très simples. Les changements de topologie sont cependant délicats à mettre à jour entre les deux modèles.

La technique que nous employons est dite des *PN-Triangles* et s'inspire d'un article de A. Vlachos [Vlachos *et al.*, 2001]. L'idée est de complexifier la surface en lui rajoutant un certain nombre de sommets dont la position est calculée en fonction des sommets déjà existants de façon à se rapprocher de la courbe de Bézier définie par ceux-ci. Les normales sont elles aussi recalculées à l'aide d'un processus particulier. On peut fixer le niveau de détail, ie. le nombre de sommets que l'on rajoute sur chaque arête.

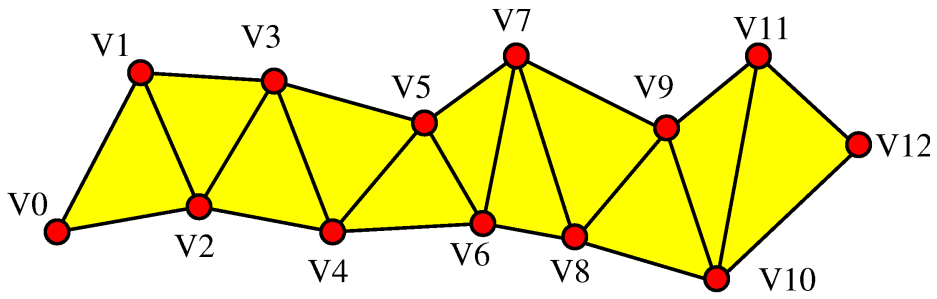


FIG. 6.10 – *L'envoi, dans l'ordre, des informations correspondant aux sommets $V0$, $V1$, ..., $V12$, soit 13 informations de sommets, définit de manière non équivoque l'ensemble des triangles en jaune. Sans cette technique, on aurait dû envoyer un triplet d'information de sommets par triangle, soit 33 informations de sommets.*

6.4.3 Bandes de triangles

Le temps passé dans les opérations graphiques se divise :

- d'une part, en le temps nécessaire à la carte pour traiter les informations et réaliser l'affichage proprement dit. Ce point requiert l'utilisation de cartes graphiques performantes afin d'assurer, par exemple, l'affichage efficace de surfaces texturées,
- d'autre part, en le temps nécessaire pour fournir à la carte les informations dont elle à besoin.

Pour chaque sommet affiché, on doit en effet transmettre à la carte huit valeurs différentes :

- la position, soit trois valeurs
- la valeur de la normale, soit trois valeurs
- les coordonnées de textures, soit deux valeurs pour les textures de surface.

Chaque triangle étant composé de trois sommets, le nombre de données transmises à la carte devient vite énorme. Nous utilisons une technique dite *des bandes de triangles*, qui divise par trois le volume de données transmises. L'idée est de ranger les triangles de la surface du maillage par bandes. Cette technique permet de définir un ensemble de n triangles avec simplement $n+2$ sommets (contre $3n$ sommets normalement, voir figure 6.10). Les bandes de triangles composant la surface du maillage sont mémorisées et doivent être redéterminées à chaque modification de la topologie de celui-ci. De plus, l'usage des bandes de triangles est facilement rendu compatible avec les PN-Triangles décrits précédemment.

Il existe une autre technique potentiellement beaucoup plus efficace que les bandes de triangles : la technique dite *des tableaux de sommets*. L'idée de cette technique est de transmettre à la carte d'une part la topologie du maillage et d'autre part des pointeurs sur des tableaux contenant les informations des sommets correspondants (position, normale, texture, etc.). Cette technique diminue très fortement le nombre d'appels de fonction car c'est la carte elle-même qui se chargera d'aller chercher les

informations dont elle à besoin. Nous avons essayé de mettre en œuvre cette technique mais cela s'est pour le moment révélé infructueux.

6.5 Conclusion

Nous avons voulu illustrer ici la diversité des problèmes qui peuvent être rencontrés lors de l'écriture d'un programme tel que le simulateur. Sans pour autant mériter l'appellation de *contributions scientifiques*, ces travaux constituent néanmoins une très grande partie du travail effectué au cours de cette thèse.

Chapitre 7

Conclusion

7.1 Rappel des travaux effectués

Le but de cette thèse était de faire évoluer le prototype du simulateur d'hépatectomie développé au sein du projet Epidaure de façon à en faire un outil crédible et pouvant éventuellement se prêter à une validation médicale. Ce travail devait s'orienter vers l'introduction des réseaux vasculaires dans le processus de simulation : modélisation, interaction avec le maillage volumique, extraction, manipulation, découpe, etc, mais aussi améliorer certaines des faiblesses des versions précédentes : réalisme de la découpe, stabilité du retour d'effort, rapidité du calcul des déformations, etc, la liste étant loin d'être limitative.

L'ensemble des travaux réalisés au cours de cette thèse a permis de répondre à cette attente :

- en développant une nouvelle structure de données de maillage efficace pour la détermination et le parcours des voisinages, caractéristiques essentielles lors du calcul des déformations (chapitre 2).
- en proposant une technique de découpe par retrait de tétraèdres qui garantit l'absence de singularité topologique dans le maillage, ce qui est important, entre autres, pour la stabilité du retour d'effort (chapitre 3),
- en fournissant une méthode de raffinement du maillage "à la volée" qui améliore la précision et le réalisme visuel de la découpe et évite d'avoir à utiliser des maillages trop fins (chapitre 3, § 3.9 et suivants).
- en s'affranchissant des hypothèses de régularité sur la surface du maillage utilisées jusqu'à présent dans le traitement des collisions,
- en améliorant la stabilité du retour d'effort en adaptant la technique dite du *modèle local* (chapitre 4),
- en introduisant un modèle de vaisseau qu'il est possible d'extraire du parenchyme et de manipuler (chapitre 5),
- en abordant certaines pistes visant à améliorer le calcul des déformations ou encore améliorant l'aspect visuel du simulateur (chapitre 6)

7.2 Limitations actuelles

Les limitations du simulateur sont encore nombreuses :

- la technique de raffinement utilisée a le défaut de multiplier rapidement le nombre de tétraèdres créés. Il faudrait explorer des méthodes non basées sur la subdivision, comme par exemple le remaillage Delaunay,
- les interactions entre l'outil et le maillage peuvent présenter quelques instabilités lorsqu'il y a plusieurs composantes connexes près de la pointe de l'outil. Le calcul des forces envoyées à l'utilisateur est géré de manière purement géométrique et il serait intéressant de dépendre plus directement des caractéristiques mécaniques du maillage,

– il n'y a pas de retour d'effort pour les vaisseaux.

7.3 Perspectives

Le prototype de simulateur auquel nous avons abouti est satisfaisant en ce qu'il a rendu crédible son utilisation effective pour l'apprentissage des techniques de laparoscopie. Un certain nombre de tâches restent cependant à accomplir :

- améliorer l'ergonomie du prototype de façon à faciliter son utilisation de la part de non informaticiens,
- créer une bibliothèque de modèles de patients différents permettant de s'entraîner sur différents cas,
- définir un certain nombre d'exercices simples et progressifs ainsi qu'une méthode permettant de les évaluer,
- introduire un certain nombre de données physiologiques pouvant intervenir au cours de la simulation (rythme cardiaque, respiration),
- améliorer l'esthétique du simulateur en modifiant d'une part le rendu des outils virtuels et utiliser la vraie géométrie des instruments (cavitron, clip, pince, ciseaux, etc.), et en introduisant un certain nombre d'effets visuels (meilleures textures, saignements, fumée) ou même sonores (bruit du cœur ou du cavitron),
- enfin, il faudrait effectuer une validation médicale qui permettrait de quantifier l'apport réel de ce simulateur pour l'apprentissage des techniques de chirurgie par cœlioscopie.

Au cours de l'année 2003 il est prévu de mettre en œuvre l'ensemble des points ci-dessus afin d'aboutir dès l'année suivante à un système directement utilisable par des élèves chirurgiens. Ces travaux se feront à l'IRCAD (Strasbourg) qui collabore depuis l'origine à la réalisation de ce prototype de simulateur.

7.4 Conclusion

A plus long terme il reste aussi plusieurs grands champs d'investigation à explorer.

- Plutôt que de représenter seul l'organe à opérer, il serait intéressant de l'inclure au milieu des organes environnants. Pour cela il faudrait arriver à modéliser efficacement les contacts entre organes afin d'éviter les auto collisions tout en conservant une résolution temps-réel.
- L'introduction des grandeurs physiologiques telles que rythme cardiaque ou la respiration serait aussi un élément crucial pour le réalisme du simulateur. Cette introduction consiste d'abord à modéliser l'influence de ces facteurs sur la simulation : par exemple le mouvement causé par le respiration, l'influence de la pression cardiaque sur la rigidité des organes ou sur le comportement des vaisseaux. Réciproquement il faut aussi modéliser l'influence de l'opération sur ces

grandeurs physiologiques : par exemple les pertes sanguines peuvent entraîner une modification du rythme cardiaque ou de la pression artérielle.

– Si le modèle mécanique utilisé pour le calcul des déformations est très réaliste, sa vitesse de convergence est insuffisante se qui conduit à un aspect relativement visqueux. De nombreuses pistes sont envisageables pour accélérer cette convergence. Il serait par exemple possible de poursuivre les travaux de C.Checoury [Checoury, 2002] visant à regrouper dans un même maillage des zones fonctionnant en pré-calculé rapides à calculer, et des zones fonctionnant en masse-tenseur plus coûteuses mais permettant la découpe. Il serait aussi possible de modifier les schémas de résolution asynchrones comme nous l'avons tenté au cours de cette thèse.

Bien qu'ils soient de plus en plus crédibles, les simulateurs de chirurgie sont en en effet encore loin d'égaler le réalisme impressionnant des simulateurs de vol. Au delà des améliorations prévisibles dues au simple mécanisme d'augmentation des performances de calcul des ordinateurs, c'est l'ouverture de ces "verrous méthodologique" qui permettra d'effectuer le saut qualitatif qui leur manque.

Bibliographie

[Aharon and Lenglet, 2002] Shmuel Aharon and Christophe Lenglet. Collision detection algorithm for deformable objects using opengl. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2489 of *LNCS*, pages 211–218, Tokyo, September 2002. Springer.

[Balaniuk and Salisbury, 2003] Remis Balaniuk and Kenneth Salisbury. Soft-tissue simulation using the radial elements method. In *International Symposium on Surgery Simulation and Soft-tissue Modeling (IS4TM)*, Juan-les-Pins, Juin 2003. Soumis.

[Balaniuk, 1999] Rémis Balaniuk. Using fast local modeling to buffer haptic data. In *Proceeding of the 4th PhantoM User Group Workshop (PUG'99)*, pages 7–11, Cambridge, October 1999.

[Bar-Cohen *et al.*, 2001] Yoseph Bar-Cohen, Constantinos Mavroidis, Mourad Bouzit, Benjamin P. Dolgin, Deborah L. Harm, George E. Kopchok, and Rodney A. White. Virtual reality robotic telesurgery simulations using memica haptic system. In Yoseph Bar-Cohen, editor, *Proceedings of SPIE Smart Structures Conference*, volume 4329, pages 357–363, March 2001.

[Baraff, 1993] D. Baraff. Non-penetrating rigid body simulation. *State of the Art Reports of EUROGRAPHICS'93, Eurographics Technical Report Series*, 1993.

[Baumann *et al.*, 1997] R Baumann, W Maeder, D Glauser, and R Clavel. The pantoscope: A spherical remote-center-of-motion parallel manipulator for force reflexion. In *IEEE Int. Conference on Robotics & Automation*, pages 718–723, 1997.

[Baur *et al.*, 1998] Charles Baur, Didier Guzzoni, and Olivier Georg. Virgy: A virtual reality and force feedback based endoscopic surgery simulator. In *Proceedings of Medicine Meets Virtual Reality (MMVR'6)*, pages 385–391, San Diego, CA, 1998.

[Beauquier et Bérard, 1994] Joffroy Beauquier et Béatrice Bérard. *Systèmes d'Exploitation : Concepts et Algorithmes*. Ediscience International, Paris, 1994.

[Benzley *et al.*, 1995] Steven E. Benzley, Ernest Perry, Brett Clark Karl Merkley, and Greg Sjaardema. Comparison of all-hexahedral and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *4th International Meshing Roundtable*, pages 179–191. Sandia National Laboratories, October 1995.

[Bøhn, 1995] Jan Helge Bøhn. Removing zero-volume parts from CAD models for layered manufacturing. *IEEE Computer Graphics & Applications*, 15(6):27–34, November 1995.

[Bielser and Gross, 2000] Daniel Bielser and Markus H. Gross. Interactive simulation of surgical cuts. In *Proc. Pacific Graphics 2000*, pages 116–125. IEEE Computer Society Press, October 2000.

[Boissonnat et Yvinec, 1995] Jean-Daniel Boissonnat et Mariette Yvinec. *Géométrie algorithmique*. Ediscience international, Paris, 1995.

[Bourguignon and Cani, 2000] David Bourguignon and Marie-Paule Cani. Controlling anisotropy in mass-spring systems. In *Computer Animation and Simulation*, pages 113–123, August 2000.

[Bourns, 2002] Bourns. *EN, Rotary Optical Encoder*. Riverside, CA, 2002.

[Boux de Casson and Laugier, 2000] F. Boux de Casson and C. Laugier. Simulating 2D tearing phenomena for interactive medical surgery simulators. In *Proc. of Computer Animation*, Philadelphia, PA (US), May 2000.

[Boux de Casson, 2000] François Boux de Casson. *Simulation dynamique de corps biologiques et changements de topologie interactifs*. Thèse de science, Université de Savoie, Chambéry (FR), Decembre 2000.

[Brouwer *et al.*, 2001] Iman Brouwer, Jeffrey Ustin, Loren Bentley, Alana Sherman, Neel Dhruv, and Frank Tendick. Measuring in vivo animal soft tissue properties for haptic modeling in surgical simulation. In J.D. Westwood et al., editor, *Medicine Meets Virtual Reality*, pages 69–74, NewPort Beach, CA, January 2001. IOS Press.

[Brown *et al.*, 2001a] Joel Brown, Kevin Montgomery, Jean-Claude Latombe, and Michael Stephanides. A microsurgery simulation system. In W.J. Niessen and M.A. Viergever, editors, *4th Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI'01)*, volume 2208 of *LNCS*, pages 137–144, Utrecht, The Netherlands, October 2001.

[Brown *et al.*, 2001b] Joel Brown, Stephen Sorkin, Cynthia Bruyns, Jean-Claude Latombe, Kevin Montgomery, and Michael Stephanides. Real-time simulation of deformable objects: Tools and application. In *Computer Animation*, Seoul, Korea, November 2001.

[Brown *et al.*, 2002] Joel Brown, Stephen Sorkina, Jean-Claude Latombe, and Kevin Montgomery. Algorithmic tools for real-time microsurgery simulation. *Medical Image Analysis*, 6(3):289–300, sep 2002.

[Brown, 1998] Peter John Cameron Brown. *Selective Mesh Refinement for Rendering*. PhD thesis, Emmanuel College, University of Cambridge, February 1998.

[Bruyns and Montgomery, 2002a] Cynthia Bruyns and Kevin Montgomery. Generalized interactions using virtual tools within the spring framework: Cutting. In James Westwood et al., editor, *Proceedings of the 10th Annual Medicine Meets Virtual Reality (MMVR'02)*, volume 85, Newport Beach, CA, January 23-26 2002. IOS Press.

[Bruyns and Montgomery, 2002b] Cynthia Bruyns and Kevin Montgomery. Generalized interactions using virtual tools within the spring framework: Probing, piercing, cauterizing, and ablating. In James Westwood et al., editor, *Proceedings of the 10th Annual Medicine Meets Virtual Reality (MMVR'02)*, volume 85, Newport Beach, CA, January 23-26 2002. IOS Press.

[Bruyns and Ottensmeyer, 2002] Cynthia Bruyns and Mark P. Ottensmeyer. Measurements of soft-tissue mechanical properties to support development of a physically based virtual animal model. In Takeyoshi Dohi and Ron Kikinis, editors, *Proc. of the IMIVA workshop of MICCAI*, volume 2488 of *LNCS*, pages 282–289, Tokyo, November 2002. Springer.

[Carter, 1998] Fiona Carter. Biomechanical testing of intra-abdominal soft tissue. In *International Workshop on Soft Tissue Deformation and Tissue Palpation*, Cambridge, MA, October 1998.

[CGAL, 2002] CGAL. *Basic Library Manuals*, May 2002. Release 2.4.

[Chabanas *et al.*, 2003] Matthieu Chabanas, Vincent Luboz, and Yohan Payan. Patient specific finite element model of the face soft tissue for computer-assisted maxillofacial surgery. *Medical Image Analysis*, 2003. (in review).

[Checoury, 2002] Cédric Checoury. Optimisation de modèles physiques de tissus mous pour la simulation chirurgicale. Rapport de DEA, Eurocom, Septembre 2002.

[Chen, 1999] Elaine Chen. Six degree-of-freedom haptic system for desktop virtual prototyping applications. In Gérard Subsol, editor, *Colloque scientifique international, réalité virtuelle et prototypage : Actes, Premières rencontres internationales de la réalité virtuelle de Laval*, pages 97–106, Laval, France, jun 1999. CNRS/Ministère de l'éducation nationale, de la recherche et de la technologie, Mairie de Laval, service communication.

[Cherqui *et al.*, 2002] Daniel Cherqui, Olivier Soubrane, Emmanuel Husson, Eric Barshasz, Olivier Vignaux, Mourad Ghimouz, Sophie Branchereau, Christophe Chardot, Frédéric Gauthier, Pierre-Louis Fagniez, and Houssin Didier. Laparoscopic living donor hepatectomy for liver transplantation in children. *The Lancet*, 359(9304):392–394, 2002.

[Clatz, 2002] Olivier Clatz. Analysis and prediction of the brain deformation during a neurosurgical procedure. Rapport de DEA mathématique vision apprentissage, École Normale Supérieure de Cachan, Septembre 2002.

[Costa and Balaniuk, 2001] Ivan Ferreira Costa and Remis Balaniuk. Lem - an approach for real time physically based soft tissue simulation. In *International Conference in Automation and Robotics (ICRA'01)*, Séoul, Corée, Mai 2001.

[Cotin *et al.*, 1999] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73, January-March 1999.

[Cotin *et al.*, 2000] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[Cotin, 1997] Stéphane Cotin. *Modèles anatomiques déformables en temps réel : Application à la simulation de chirurgie avec retour d'effort.* Thèse de sciences, Université de Nice Sophia-Antipolis, Novembre 1997.

[Cover *et al.*, 1993] S. A. Cover, N. F. Ezquerra, and J. F. O'Brien. Interactively Deformable Models for Surgery Simulation. *IEEE Computer Graphics and Applications*, pages 68–75, 1993.

[Coxeter, 1934] H Coxeter. Discrete groups generated by reflexions. *Annals of Mathematics*, 16:13–29, 1934.

[Dan, 1999] Diane Dan. *Caractérisation mécanique du foie humain en situation de choc*. Thèse de biomécanique, Université Paris VII, 27 Septembre 1999.

[Danovaro, 2001] Emanuele Danovaro. Modeling and processing large geometrical datasets: from meshes to compact multiresolution models, December 2001. Thesis Proposal, Universita di Genova.

[Davanne *et al.*, 2002] J Davanne, P. Meseure, and C. Chaillou. Stable haptic interaction in a dynamic virtual environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, pages 2881–2886, Lausanne, sep 2002.

[Debunne *et al.*, 2001] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Computer Graphics Proceedings*, August 2001.

[Debunne, 2000] Gilles Debunne. *Animation multirésolution d'objets déformables en temps-réel, Application à la simulation chirurgicale*. Thèse de sciences, Institut National Polytechnique de Grenoble, Décembre 2000.

[Deguet *et al.*, 1998] A. Deguet, A. Joukhadar, and C. Laugier. Models and algorithms for the collision of rigid and deformable bodies. In P. K. Agarwal, L. E. Kavraki, and M. T. Mason, editors, *Robotics: the algorithmic perspective*, pages 327–338. AKPeters, 1998. Proc. of the Workshop on the Algorithmic Foundations of Robotics. Houston, TX (US). March 1998.

[Delaunay, 1934] Boris N. Delaunay. Sur la sphère vide. *Bulletin de l'Academie des Sciences de L'URSS, Classe des Sciences Mathématiques et Naturelles*, 7:793–800, 1934.

[Despopoulos et Silbernagl, 1999] Agamemnon. Despopoulos et Stefan Silbernagl. *Atlas de poche de physiologie*. Médecine-Sciences. Flammarion, 1999.

[Digital Corporation, 2002] US Digital Corporation. *Quick Assembly Two and Three Channel Optical Encoders*, 2002.

[Dompierre *et al.*, 1998] J. Dompierre, P. Labbé, F. Guibault, and R. Camarero. Proposal of benchmarks for 3D unstructured tetrahedral mesh optimization. In *7th International Meshing Roundtable*, pages 459–478, Dearborn, MI, October 1998.

[Désidéri, 1998] Jean-Antoine Désidéri. *Modèles discrets et schémas itératifs, application aux algorithmes multigrilles et multidomaines*. Hermes, Paris, 1998.

[Ellis *et al.*, 1997] R. E. Ellis, O. M. Ismaeil, and M. Lipsett. Design and evaluation of a high-performance haptic interface. *Robotica*, 14:321–327, 1997.

[Forest *et al.*, 2002a] Clément Forest, Hervé Delingette, and Nicholas Ayache. Cutting simulation of manifold volumetric meshes. In *Modeling & Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, 2002.

[Forest *et al.*, 2002b] Clément Forest, Hervé Delingette, and Nicholas Ayache. Cutting simulation of manifold volumetric meshes. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2489 of *LNCS*, pages 235–244, Tokyo, September 2002. Springer.

[Forest *et al.*, 2002c] Clément Forest, Hervé Delingette, and Nicholas Ayache. Removing tetrahedra from a manifold mesh. In *Computer Animation (CA'02)*, pages 225–229, Geneva, Switzerland, June 2002. IEEE Computer Society.

[France *et al.*, 2002] Laure France, Alexis Angelidis, Philippe Meseure, Marie-Paule Cani, Julien Lenoir, François Faure, and Christophe Chaillou. Implicit representations of the human intestines for surgery simulation. In *Conference on Modeling and Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, Rocquencourt, Novembre 2002.

[Frey et George, 1999] Pascal Jean Frey et Paul-Louis George. *Maillages, applications aux éléments finis*. Hermes Sciencs, Paris, 1999.

[Frey, 1993] Pascal Frey. *Génération automatique de maillages*. Thèse de sciences, Université Louis Pasteur de Strasbourg, Novembre 1993.

[Fung, 1993] Yuan-Cheng Fung. *Biomechanics, Mechanical Properties of Livig Tissues*. Springer, second edition, 1993.

[GAMHIC3D, 2000] GAMHIC3D. *Adaptive Tetrahedral Mesh Generator*. INRIA-Simulog, feb 2000.

[Ganovelli *et al.*, 2000] Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. *Computer Graphics Forum (Eurographics 2000 Conference Proc.),*, 19(3):271–282, 2000.

[George and Borouchaki, 1997] Paul-Louis George and Houman Borouchaki. *Triangulation de Delaunay et maillage, Applications aux éléments finis*. Hermes, Paris, 1997.

[GHS3D, 1999] GHS3D. *Tetrahedral Mesh Generator*. INRIA-Simulog, may 1999.

[Gibson *et al.*, 1997] S. Gibson, J. Samosky, A. Mor, C. Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, and N. McKenzie. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback . In J. Troccaz, E. Grimson, and R. Mosges, editors, *Proceedings of the First Joint Conference CVRMed-MRCAS'97*, volume 1205 of *Lecture Notes in Computer Science*, pages 369–378, March 1997.

[Gosselin, 2000] Florian Gosselin. *Développement d'outils d'aide à la conception d'organes de commande pour la téléopération à retour d'effort*. Thèse de sciences fondamentales appliquées, Université de Poitiers, Paris, Juin 2000.

[Gottschalk *et al.*, 1996] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[Guéziec *et al.*, 2001] André Guéziec, Gabriel Taubin, Francis Lazarus, and Bill Horn. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics*, pages 136–151, 2001.

[Henle, 1979] Michael Henle. *A Combinatorial introduction to topology*. Dover,1994, 1979.

[Hoff *et al.*, 2002] Kenneth E. Hoff, Andrew Zaferakis, Ming Lin, and Dinesh Manocha. Fast 3d geometric proximity queries between rigid and deformable models

using graphics hardware acceleration. Technical report, University of North Carolina, Computer Science, 2002.

[Hoffmann, 1989] Christoph M. Hoffmann. *Geometric and solid modeling: an introduction*. Morgan Kaufmann Publishers Inc., 1989.

[Hollands and Trowbridge, 1996] Robin Hollands and E A Trowbridge. A PC-based virtual reality arthroscopic surgical trainer. In *Proc. Simulation in Synthetic Environments*, volume 28(2), pages 17–22, New Orleans, apr 1996. The Society for Computer Simulation.

[Hubbard, 1996] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[Ibañez *et al.*, 2003] Luis Ibañez, Will Schroeder, and the Insight Consortium. *The ITK Software Guide*, March 2003.

[Imagis, 2001] Projet Imagis. Rapport d'activité, INRIA, 2001.

[J. Kaye, 1998] D. Metaxas J. Kaye, F. Primiano. A 3d virtual environment for modeling mechanical cardiopulmonary interactions. *Medical Image Analysis (Media)*, 2(2):169–195, June 1998.

[James and Pai, 1999] Doug L. James and Dinesh K. Pai. Artdefo: Accurate real time deformable objects. In *Proceedings of the 26th Annual Conference on Computer Graphics and (SIGGRAPH'99)*, pages 65–72, August 1999.

[Kataoka *et al.*, 2002] Hiroyuki Kataoka, Toshikatsu Washio, Kiyoyuki Chinzei, Kazuyuki Mizuhara, Christina Simone, and Allison M. Okamura. Measurement of the tip and friction force acting on a needle during penetration. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2488 of *LNCS*, pages 216–223, Tokyo, September 2002. Springer.

[Kauer and Vuskovic, 1999] M Kauer and V Vuskovic. In vivo measurment of elastomechanical properties of soft biological tissues, 1999.

[Keeve *et al.*, 1996] Erwin Keeve, Sabine Girod, and Bernd Girod. Cranofacial surgery simulation. In 4^{th} *Visualization in Biomedical Computing, VBC'96*, pages 541–546, Hamburg, Germany, September 1996.

[Kheddar, 2002] Abderrahmane Kheddar. *Téléopération et Robotique*, chapitre 4. Rendus haptiques. IC2, Systèmes Automatisés. Hermes, 2002.

[Kühnapfel *et al.*, 2001] U. Kühnapfel, H.K. Çakmak, H. Maaß, and S. Waldhausen. Models for simulating instrument-tissue interactions. Tutorial of the 9^{th} Medicine Meets Virtual Reality (MMVR 2001), Jan 2001. http://www-kismet.iai.fzk.de/-KISMET/kispub.html.

[Krishnan and Manocha, 1996] S. Krishnan and D. Manocha. Efficient representations and techniques for computing b-rep's of csg models with nurbs primitives. In *CSG'96 Set-theoretic Solid Modeling Techniques and Applications, Information Geometers*, pages 101–122. Winchester, April 1996.

[Krissian, 2000] Karl Krissian. *Traitement multi-échelle : Applications à l'imagerie médicale et à la détection tridimensionnelle de vaisseaux*. Thèse de sciences, université de Nice Sophia-Antipolis, janvier 2000.

[Lamy and Chaillou, 1999] Dominique Lamy and Christophe Chaillou. Design, implementation and evaluation of an haptic interface for surgical gestures training. In Gérard Subsol, editor, *Colloque scientifique international, réalité virtuelle et prototypage : Actes, Premières rencontres internationales de la réalité virtuelle de Laval*, pages 107–116, Laval, France, jun 1999. CNRS/Ministère de l'éducation nationale, de la recherche et de la technologie, Mairie de Laval, service communication.

[Lefevre, 1994] A. Lefevre. Foie et pancréas : la chimie fine. *Science et vie*, H.S. 187:96–105, Juin 1994.

[Lenoir *et al.*, 2002] Julien Lenoir, Philippe Meseure, Laurant Grisoni, and Christophe Chaillou. Surgical thread simulation. In *Conference on Modeling and Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, Rocquencourt, Novembre 2002.

[Löhner and Baum, 1992] Rainald Löhner and Joseph D. Baum. Adaptative h-refinement on 3D unstructured grids for transient problems. *International Journal for Numerical Methods in Fluids*, 14(12):1407–1419, December 1992.

[Lin, 1993] Ming C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1993.

[Liu and Joe, 1994] Anwei Liu and Barry Joe. Relation between tetrahedron shape measures. *BIT*, 34(2):268–287, 1994.

[Liu and Joe, 1995] Anwei Liu and Barry Joe. Quality local refinement of tetrahedral meshes based on bisection. *SIAM Journal on Scientific Computing*, 16(6):1269–1291, nov 1995.

[Liu and Joe, 1996] Anwei Liu and Barry Joe. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *Mathematics of Computation*, 65(215):1183–1200, 1996.

[Lombardo *et al.*, 1999] Jean-Christophe Lombardo, Marie-Paule Cani, and Fabrice Neyret. Real-time collision detection for virtual surgery. In *Computer Animation*, Geneva Switzerland, May 26-28 1999.

[Maaß and Kühnapfel, 1999] H. Maaß and U. Kühnapfel. Noninvasive measurement of elastic properties of living tissue. In *Congress on Computer Assisted Radiology and Surgery (CARS '99)*, pages 865–870, Paris, June 23–26 1999.

[Maaß *et al.*, 2003] Heiko Maaß, Benjamin Chantier, Hüseyin Cakmak, and Uwe Kuehnapfel. How to add force feedback to a surgery simulator. In *International Symposium on Surgery Simulation and Soft-tissue Modeling (IS4TM)*, Juan-les-Pins, Juin 2003. Soumis.

[Marescaux *et al.*, 1998] J. Marescaux, J-M. Clément, V. Tassetti, C. Koehl, S. Cotin, Y. Russier, D. Mutter, H. Delingette, and N. Ayache. Virtual reality applied to hepatic surgery simulation : The next revolution. *Annals of Surgery*, 228(5):627–634, November 1998.

[Mark *et al.*, 1996] William R. Mark, Scott C. Randolph, Mark Finch, James M. Van Verth, and II Russell M. Taylor. Adding force feedback to graphics systems: issues and solutions. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 447–452. ACM Press, 1996.

[Massie and Salisury, 1994] T.H. Massie and J.K. Salisury. The PHANTOM haptic interface: a device for probing virtual objects. In *Proceedings of the ASME Winter Annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems*, Chicago, IL, nov 1994.

[Maurel *et al.*, 1991] W. Maurel, Y. Wu, N. Magnenat Thalmann, and D. Thalmann. *Biomedichanical Models for Soft Tissue Simulation*. ESPRIT Basic Research Series. Springer, Berlin, 1991.

[Maxon Motor ag, 2002] Maxon Motor ag. Catalogue Maxon DC Motor, Avril 2002.

[Mazzella, 1999] Frédéric Mazzella. Auto acquisition of elastic properties of soft tissues for surgical simulation. Stage de recherche mip, École Normale Supérieure, Stanford, CA, Juillet 1999.

[Mendoza *et al.*, 2001] Cesar Mendoza, Christian Laugier, and François Boux de Casson. Virtual reality cutting phenomena using force feedback for surgery simulations. In *Proc. of the IMIVA workshop of MICCAI*, Utrecht(NL), November 2001.

[Mendoza Serrano and Laugier, 2001] Cesar Mendoza Serrano and Christian Laugier. Realistic haptic rendering for highly deformable virtual objects. In *Proc. of the Int. Conf. on Virtual Reality*, Yokohama (JP), March 2001.

[Meseure *et al.*, 1995] Philippe Meseure, Sylvain Karpf, Christophe Chaillou, Patrick Dubois, and Jean-François Rouland. Low-cost medical simulation: a retinal laser photocoagulation simulator. In *Proceedings of the Graphics'Interface*, pages 155–162, Québec, 17–19 May 1995.

[Michael *et al.*, 1997] Dinsmore Michael, Noshir A. Langrana, Grigore Burdea, and Kemi Ladeji. Virtual reality training simulation for palpation of subsurface tumors. In *IEEE International Symposium on Virtual Reality and Applications (VRAIS'97)*, pages 54–60, Albuquerque, NM, March 1997.

[Mäntylä, 1987] Martti Mäntylä. *An introduction to solid modeling*. Computer Science Press Inc., Rockville MA, 1987.

[Montgomery *et al.*, 2001] Kevin Montgomery, Cynthia Bruyns, Simon Wildermuth, LeRoy Heinrichs, Christopher Hasser, Stephanie Ozenne, and David Bailey. Surgical simulator for operative hysteroscopy. In *Proceedings of the 12th IEEE Visualization Conference (VIS'01)*, San Diego, CA, October 2001.

[Moore and Warren, 1990] D Moore and J Warren. Adaptive mesh generation I: Packing space. Technical report, Department of Computer Science, Rice University, 1990.

[Mor and Kanade, 2000] Andrew B. Mor and Takeo Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *MICCAI*, pages 598–607, 2000.

[Nakao *et al.*, 2002] Megumi Nakao, Tomohiro Kuroda, Hiroshi Oyama, Masaru Komori, Tetsuya Matsuda, and Takashi Takahashi. Combining volumetric soft tissue cuts for interventional surgery simulation. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2489 of *LNCS*, pages 178–185, Tokyo, September 2002. Springer.

[Neyret *et al.*, 2002] Fabrice Neyret, Raphael Heiss, and Franck Senegas. Realistic

rendering of an organ surface in real-time for laparoscopic surgery simulation. *the Visual Computer*, 18(3):135–149, may 2002.

[Nienhuys and van der Stappen, 2001] Han-Wen Nienhuys and A. Frank van der Stappen. Supporting cuts and finite element deformation in interactive surgery simulation. In W.J. Niessen and M.A. Viergever, editors, *Procs. of the Fourth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'01)*, pages 145–152. Springer Verlag, October 2001.

[Nienhuys and van der Stappen, 2002] Han-Wen Nienhuys and A. Frank van der Stappen. Interactive cutting in triangulated surfaces, a delaunay approach. In *Procs. of the Fifth International Workshop on Algorithmic Foundations of Robotics (WAFR'02)*, December 2002.

[Norton *et al.*, 1991] Alan Norton, Greg Turk, Bob Bacon, John Gerth, and Paula Sweeney. Animation of fracture by physical modeling. *The Visual Computer: International Journal of Computer Graphics*, 7(4):210–219, 1991.

[O'Brien and Hodgins, 2000] James F. O'Brien and Jessica K. Hodgins. Animating fracture. *Communications of the ACM*, 43(7):68–75, 2000.

[Olufsen, 1998] Mette Sofie Olufsen. *Modeling the Arterial System with Reference to an Anesthesia Simulator*. PhD thesis, Roskilde University, Department of Mathematics, Denmark, May 1998.

[O'Sullivan and Dingliana, 1999] C. O'Sullivan and J Dingliana. Real-time collision detection and response using sphere-trees. In *Proceedings of the 15th Spring Conference on Computer Graphics*, pages 83–92, Budmerice, Slovakia, April 1999.

[Ottensmeyer and Salisbury, 2001] Mark P. Ottensmeyer and John Kenneth Salisbury. In vivo data acquisition instrument for solid organ mechanical property measurement. In Wiro J. Niessen and Max A. Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001, 4th International Conference, Utrecht, The Netherlands, October 14-17, 2001, Proceedings*, volume 2208 of *Lecture Notes in Computer Science*, pages 975–982. Springer, 2001.

[Ousterhout, 1994] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[Paloc *et al.*, 2002] Celine Paloc, Fernando Bello, Richard I. Kitney, and Ara Darzi. Online multiresolution volumetric mass spring model for real time soft tissue deformation. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2489 of *LNCS*, pages 219–226, Tokyo, September 2002. Springer.

[Papadopoulos *et al.*, 2002] Enangelos Papadopoulos, Kostas Vlachos, and Dionyssios Mitropoulos. On the design of a low-force 5-dof force-feedback haptic mechanism. In *Proceedings of Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC'02)*, Montreal, September 2002.

[Persiano *et al.*, 1993] R. Persiano, J. Comba, and V. Barbalho. An adaptive triangulation refinement scheme and construction. In *Proceedings of the VI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, Recife, Brazil, October 1993.

[Perucchio *et al.*, 1989] Renato Perucchio, Saxena Makul, and Kela Ajay. Automatic mesh generation from solid models based on recursive spatial decompositions. *International Journal For Numerical Methods In Engineering*, 28(11):2469–2501, nov 1989.

[Petersik *et al.*, 2003] Andreas Petersik, Bernhard Pflesser, Ulf Tiede, Karl-Heiz Höhne, and Rudolf Leuwer. Realistic haptic interaction in volume sculpting for surgery simulation. In *International Symposium on Surgery Simulation and Soft-tissue Modeling (IS4TM)*, Juan-les-Pins, Juin 2003. Soumis.

[Picinbono and Lombardo, 1999] Guillaume Picinbono and Jean-Christophe Lombardo. Extrapolation: a solution for force feedback? In *International Scientific Workshop on Virtual Reality and Prototyping*, pages 117–125, Laval France, June 3-4 1999.

[Picinbono *et al.*, 2000] G. Picinbono, H. Delingette, and N. Ayache. Real-Time Large Displacement Elasticity for Surgery Simulation: Non-Linear Tensor-Mass Model. In *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery: MICCAI 2000*, pages 643–652, October 2000.

[Picinbono *et al.*, 2001a] G. Picinbono, Jean-Christophe Lombardo, Hervé Delingette, and Nicholas Ayache. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. *Journal of Visualisation and Computer Animation*, 13(3):147–167, 2001.

[Picinbono *et al.*, 2001b] Guillaume Picinbono, Hervé Delingette, and Nicholas Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference Robotics and Automation*, Seoul Korea, May 2001. Best conference paper award.

[Picinbono *et al.*, 2002] G. Picinbono, H. Delingette, and N. Ayache. Non-Linear Anisotropic Elasticity for Real-Time Surgery Simulation. *Graphical Models*, 2002. In press.

[Picinbono, 2001] Guillaume Picinbono. *Modèles géométriques et physiques pour la simulation d'interventions chirurgicales*. Thèse de sciences pour l'ingénieur, Université de Nice Sophia-Antipolis, Février 2001.

[Péquignot, 1999] Julien Péquignot. Optimisation géométrique locale de maillages tétraédriques. Technical report, INRIA, 1999. Rapport de projet ESSI3 - CSI.

[Qin, 2000] Hong Qin. Fem-based dynamic subdivision surfaces. *8th Pacific Conference on Computer Graphics and Applications*, pages 184–191, October 2000. ISBN 0-7695-0868-5.

[Rhodin, 1980] Johannes A.G. Rhodin. *Handobook of Physiology*, volume II, chapter 2. The cardiovascular system, pages 1–31. American Physiology Society, Bethesda, MD, 1980.

[Riwan, 2002] Alain Riwan. *Téléopération et Robotique*, chapitre 3. Actionneurs et motorisations. IC2, Systèmes Automatisés. Hermes, 2002.

[Rubini, 1998] Alessandro Rubini. *Linux Device Drivers*. O'Reilly & Associates Inc., Sebastopol, CA, 1^{st} edition, fev 1998.

[Ruspini *et al.*, 1997] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The

haptic display of complex graphical environments. In *Computer Graphics (SIGGRAPH 97 Conference Proceedings)*, pages 345–352. ACM SIGGRAPH, 1997.

[Sakuma *et al.*, 2003] Ichiro Sakuma, Yosuke Nishimura, Chee Kong Chui, Etsuko Kobayashi, Hiroshi Inada, Xian Chen, and Toshiaki Hishada. In vitro measurement of mechanical properties of liver tissue under compression and elongation using a new test piece holding method with surgical glue. In *International Symposium on Surgery Simulation and Soft-tissue Modeling (IS4TM)*, Juan-les-Pins, Juin 2003. Soumis.

[Sanna and Montuschi, 1998] Andrea Sanna and Paolo Montuschi. An efficient algorithm for ray casting of CSG animation frames. *The Journal of Visualization and Computer Animation*, 9(4):229–242, 1998.

[Satava, 1996] Richard Satava. The current status of the future. In *Proc. of 4th Conf. Medicine Meets Virtual Reality (MMVR'96)*, Interactive Technology and the New Paradigm for He althcare, pages 100–106. IOS Press, January 1996.

[Schijven and Jakimowicz, 2002] Marlies Schijven and Jack Jakimowicz. Face-, expert, and referent validity of the xitact ls500 laparoscopy simulator. *Surgical Endoscopy*, July 2002. electronic publication.

[Schwartz *et al.*, 2002] Jean-Marc Schwartz, Marc Dellinger, Rancourt. Denis, Christian Moisan, and Denis Laurendeau. Modeling liver tissue properties using a non-linear viscoelastic model for surgery simulation. In *Conference on Modeling and Simulation for Computer-aided Medicine and Surgery (MS4CMS'02)*, Rocquencourt, Novembre 2002.

[Selle *et al.*, 2002] Dirk Selle, Bernhard Preim, Andrea Schenk, and Heinz Otto Peitgen. Analysis of vasculature for liver surgery planning. *IEEE Transaction on Medical Imaging*, 21(8), August 2002.

[Sen, 1997] Sensable Technologies Inc. *GHOST Software Developer's Toolkit*, 1997.

[Sermesant *et al.*, 2002] Maxime Sermesant, Clément Forest, Xavier Pennec, Hervé Delingette, and Nicholas Ayache. Biomechanical model construction from different modalities: Application to cardiac images. In Takeyoshi Dohi and Ron Kikinis, editors, *Medical Image Computing and Computer-Assisted Intervention (MICCAI'02)*, volume 2488 of *LNCS*, pages 714–721, Tokyo, September 2002. Springer.

[Sermesant, 2003] Maxime Sermesant. *Modèle électromécanique du cœur pour l'analyse d'image et la simulation.* Thèse de sciences de l'ingénieur, Université de Sophia Antipolis, 2003.

[Shewchuk, 2000] Jonathan Richard Shewchuk. Mesh generation for domains with small angles. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 1–10. ACM Press, 2000.

[Soler, 1998] Luc Soler. *Une nouvelle méthode de segmentation des structures anatomiques et pathologiques : application aux angioscanners 3D du foie pour la planification chirurgicale.* Thèse de science, Université d'Orsay, Novembre 1998.

[Srinivasan *et al.*, 1997] Rajagopalan Srinivasan, Shiaofen Fang, and Su Huang. Volume rendering by templatebased octree projection. In *Workshop Eurographics 97. Visualization in Scientific Computing.*, 1997.

[Stepanov and Lee, 1994] A. A. Stepanov and M. Lee. The Standard Template Library. Technical Report X3J16/94-0095, WG21/N0482, Hewlett-Packard Laboratories, July 1994.

[Swan, 1993] J. Edward Swan. Octree-based collision detection with fast neighbor finding. Technical report, Ohio State University, December 1993.

[Székely *et al.*, 1999] G. Székely, M.Baijka, and C. Brechbuhler. Virtual reality based simulation for endoscopic gynaecology. In *proceedings of Medicine Meets Virtual Reality (MMVR'99)*, pages 351–357, San Francisco (USA), 1999.

[Székely *et al.*, 2000] G. Székely, Ch. Brechbühler, J. Dual, R. Enzler, J. Hug, R. Hutter, N. Ironmonger, M. Kauer, V. Meier, P. Niederer, A. Rhomberg, P. Schmid, G. Schweitzer, M. Thaler, V. Vuskovic, and G. Tröster. Virtual reality-based simulation of endoscopic surgery. *Presence*, 9(3):310–333, 2000.

[Taubin *et al.*, 1998] G. Taubin, W.P Horn, F. Lazarus, and J. Rossignac. Geometry coding and VRML. *Proceedings of the IEEE*, 86(6):1228–1243, June 1998.

[Taylor *et al.*, 1993] Russell M. Taylor, Warren Robinett, Vernon L. Chi, Jr. Frederick P. Brooks, William V. Wright, R. Stanley Williams, and Erik J. Snyder. The nanomanipulator: a virtual-reality interface for a scanning tunneling microscope. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 127–134. ACM Press, 1993.

[Teillaud, 1999] Monique Teillaud. 3D triangulations in CGAL. In Hervé Brönnimann, editor, *15th European Workshop on Computational Geometry*, pages 175–178, Juan les Pins, 1999. INRIA.

[Tendick *et al.*, 2000] Frank Tendick, Michael Downes, Tolga Goktekin, Murant Cenk Cavusoglu, Danvid Feygin, Xuniei Wu, Roy Eyal, Mary Hegarty, and Lawrence W. Way. A virtual environment testbed for training laparoscopic surgical skills. *Presence*, 9(3):236–255, June 2000.

[Tonet, 2002] Oliver Tonet. *Computer Assistance for Knee Arthroscopy and Boyond, From Augmented Reality to Surgery Training.* PhD thesis, Scuola Superiore Sant'Anna, Pisa, Italia, February 2002.

[Tor, 2003a] Lecture on circulatory system, 2003. University of Toronto, Department of Human Biology, Program of Health and Disease, Course of Vertebrate Histology and Histopathology, `http://http://hmb.utoronto.ca/HMB302H/`.

[Tor, 2003b] Lecture on liver and gall bladder, 2003. University of Toronto, Department of Human Biology, Program of Health and Disease, Course of Vertebrate Histology and Histopathology, `http://http://hmb.utoronto.ca/HMB302H/`.

[van den Bergen, 1997] Gino van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–14, 1997.

[Verdonk, 1996] B Verdonk. *Segmentation, mesure et visualisation des vaisseaux sanguins à partir d'angiographies 3D par résonnances.* Thèse de science, ENST, Octobre 1996.

[Verma and Gelder, 1998] Vivek Verma and Allen Van Gelder. Decimation of tetrahedral grids with error control. Technical Report UCSC-CRL-97-25, University of California, Santa Cruz, June 1998.

[Vlachos *et al.*, 2001] A. Vlachos, J. Peters, C. Boyd, and J. Mitchell. Curved PN triangles. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, August 2001.

[Ward *et al.*, 1999] James Ward, Kevin Sherman, Derek Wills, Amr Mohsen, and Martin Crawshaw. The acquisition of force feedback data for a virtual environment knee arthroscopy training system. In *Virtual Reality and 3D Modeling Symposium, Advanced Simulation Technologies Conference (ASTC '99)*, pages 87–92, San Diego, CA, April 87–92 1999.

[Windecker *et al.*, 2001] Stephan Windecker, Isabella Mayer, Gabriella De Pasquale, Willibald Maier, Olaf Dirsch, Philip De Groot, Ya-Ping Wu, Georg Noll, Boris Leskosek, Bernhard Meier, and Otto M. Hess. Stent coating with titanium-nitride-oxide for reduction of neointimal hyperplasia. *Circulation*, 104:928–933, 2001.

[Witkin *et al.*, 1994] Andrew Witkin, David Baraff, and Michael Kass. An introduction to physically based modeling, 1994. SIGGRAPH'94 Course Notes, Course No. 32.

[Woo *et al.*, 1997] Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programing Guide*. Addison-Wesley, 1997.

[Woo *et al.*, 1998] Ki Young Woo, Byoung Dae Jin, and Kwon Dong-Soo. A 6 dof force-reflecting hand controller using the fivebar parallel mechanism. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 16–20, Leuven, Belgium, May 1998.

[Wu *et al.*, 2001] Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 349–358. Blackwell Publishing, 2001.

[Yao and Taylor, 2000] Jianhua Yao and Russel Taylor. Tetrahedral mesh modeling of density data for anatomical atlases and intensity-based registration. In A.M. DiGioia and S. Delp, editors, *Third International Conference on Medical Robotics, Imaging And Computer Assisted Surgery (MICCAI 2000)*, volume 1935 of *Lectures Notes in Computer Science*, pages 531–540, Pittsburgh, Pennsylvanie USA, october 11-14 2000. Springer.

[Youngblut *et al.*, 1996] Christine Youngblut, Rob E. Johnson, Sarah H. Nash, Ruth A. Wienclaw, and Craig A. Will. Review of virtual environment interface technology. IDA Paper P-3186, Institute for Defense Analyses, Alexandria, Virginia, march 1996. http://www.hitl.washington.edu/scivw/IDA/.

[Zilles and Salisbury, 1995] Craig B. Zilles and John Kenneth Salisbury. A constraint-based god-object method for haptic display. In *International Conference on Intelligent Robots and Systems*, volume 3, pages 146–151, Pittsburgh, Pennsylvania, 1995.